UNIVERSITY OF CALGARY

STRATOS: The Design of Visualization to Support Decision-making in Software Release Planning

by

Bon Adriel Aseniero

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA

DECEMBER, 2014

© Bon Adriel Aseniero 2014

ABSTRACT

Software is typically developed in incremental stages or *releases*. Planning releases involves deciding on which features of the software should have implementation priority. This is a complex planning process involving numerous constraints and factors, *trade-offs*, that often make decisions difficult. Since the success of a product depends on this plan, it is vital for planners to examine the trade-offs between different alternatives in order to make an informed choice. To support this type of decision-making, my exploration involved designing and implementing STRATOS—a visualization tool showing several software release plans simultaneously within a singular layout, helping planners understand the differences among them. Through a qualitative evaluation, I found that it enabled a range of decision-making processes, ultimately helping participants in choosing an optimal release plan. My contributions include the hybrid visualization, STRATOS, and the findings from its evaluation that implicate design for future visualizations supporting decision-making.

ACKNOWLEDGEMENTS

This thesis would not have been possible without the help and encouragement I have received from a number of people. Most importantly, I would like to express my deepest gratitude to my supervisors, Dr. Sheelagh Carpendale and Dr. Anthony Tang, who have given me much needed academic guidance and support throughout my master's degree. I thank them for acting as my mentors—inspiring me with their wonderful insights, and helping me grow both academically and personally through countless advice. I also express the same gratitude to Dr. Guenther Ruhe who have provided me with guidance and imparting knowledge on software release planning—knowledge that prove necessary in the completion of this thesis. I also thank Dr. Ehud Sharlin who spearheaded my interest in Human–Computer Interaction, and Dr. Saul Greenberg for the advice and encouragement he has given me. I also thank my examiners, Dr. Larry Katz and Dr. Robert Kremer.

I would also like to thank the Department of Computer Science at the University of Calgary, and to Alberta Innovates Technology Futures (AITF) and Graphics Animation and New Media (GRAND) whose funding allowed me to pursue my research in information visualization.

I also express my gratitude to all of the members of the Interactions Lab who have helped me throughout the course of my stay in the lab. I consider these people to be both colleagues and friends who helped me grow as an individual: I give thanks to David Ledo and Tiffany Wun whose collaborative contributions enormously helped in the completion of this thesis; to Lindsay MacDonald, Brennan Jones, and Jennifer Payne who helped me in making this

thesis more pleasant to read; and to Jagoda Walny, Setareh Aghel Manesh, Jiannan Li, Mona Hosseinkhani, Fateme Rajabi, and Claudia Maurer who helped in every appreciable way.

I give my sincere thanks to special friends who have provided me with much needed support and encouragement: Riah Fielding-Walters, Jonny Hu, Javier Lopez-Montenegro Ramil, Riane Vardeleon, Leo Leung, Derek Szeto, and Anthony Yan. Thanks as well to all the members of Eidolon Free Company who shared their time with me, giving valued support and entertainment: Len Tan, Jasmine Tan, and Natalee Koh.

Lastly, I wish to express my deepest gratitude to my family: to my parents, Purificacion Aseniero and Aniano Aseniero, whose countless support, hard work, and sacrifice helped me become successful, and to my sister, Faye Ann Aseniero, who helped me improve through her own way–Thank you for believing in me!

To Mama and Papa

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1 Figures

Chapter 2 Figures

Chapter 3 Figures

Chapter 4 Figures

Chapter 5 Figures

Appendix II Sketches and Iterations

LIST OF TABLES AND CODE LISTINGS

LIST OF ABBREVIATIONS

2D ……...……...… Two-dimensional

FGC ……. Feature Growth Chart

FSC …… Feature Survival Chart

HCI ……..…… Human–Computer Interaction

HTML5 ……..…..…….. HyperText Markup Language 5

Infovis …… Information Visualization

SHFP …….…….. Stakeholder Feature Points

STRATOS ……..... Strategic software release planning Oversight Support

UML …….. Unified Modelling Language

Chapter 1

INTRODUCTION

This thesis explores how *planners'* (e.g. project or product managers, development teams, etc.) decision-making processes can be assisted in order to enable them to make informed decisions. A well-informed decision is vital in choosing a plan for releasing software into market. Normally, this is done through meticulous examination of different factors and constraints that are typically interconnected with one another. This thesis is concerned with ***how to support planners in choosing an optimal plan by visualizing the interrelated factors of software release planning.***

In this chapter, I give background information about the practice of software release planning itself, and the importance of decision-making process in this practice. I then outline my motivation and research, using visualization as a supporting tool in this process, and articulate the research scope of my thesis. Finally, I give an overview of the structure of this thesis document and its subsequent chapters.

1.1. Background and Motivation

Varying models of software development are used in industry, including iterative and incremental practices, as well as newer agile methodologies. Companies that are trying to deliver a product work under several constraints (e.g. time, budget, personpower), and often have to contend with fluctuating and growing sets of customer requirements. Thus, it is important for large projects to make effective and efficient decisions about the use of resources—that is, deciding on a development plan to follow: what order features should be developed, which features should be postponed, how resources should be divided, etc. The process of structuring and managing project plans to balance between factors such as *stakeholder satisfaction, resource allocation, feature dependencies, etc.* is known as *release planning* (Amandeep, Ruhe and Standford 2004).

1.1.1. Common Practices in Release Planning

Common practices in release planning include the assessment of plans, resources, releases, features, and stakeholders. A *plan* (also referred to as an *alternative* when referring to a plan in a solution set containing multiple plans) contains the prioritization of features, the timing of releases, and the allocation of resources. *Features* are the different components or capabilities of the software being released. Each alternative contains a subset of features grouped into *releases* representing cycles of development, and requires certain maximum amount of *resources* (e.g. budget, hours of labour, risk, etc.) defined by the planner which are then allocated into each release phase as needed for the implementation of features (Ruhe 2011). This allocation can be changed to adapt to the needs of the plan. *Stakeholders*, in broad terms, are people who have a vested interest in the project which include but not limited to company officials, members of the development team, to the expected customers

or end-users. They vote on feature importance in terms of their priority and their perceived risk.

The goal of release planning is to find an optimal release plan that balances these factors. To approach optimality, the decision process must consider conditions that involve resource constraints and non-trivial properties (e.g. adhering to the core values of the project stakeholders), which can quickly grow in complexity with even just a few components (Jantunen, et al. 2011). Human involvement and analysis is therefore required in order to reach a final decision. For example, release planning methods like EVOLVE II are meant to be used by project managers (Greer and Ruhe 2004). EVOLVE II consists of a cycle of modelling, exploration, and consolidation. This approach analyses different releases, comparing multiple possible arrangements of features based on satisfaction outcomes. The result is a solution set of algorithmically optimized alternatives. In this method, a human planner still chooses the best plan within the solution set. To achieve good results, planners must have a good understanding of the project. In order to gain this knowledge, Amandeep et al. outlined the six steps comprising release planning (Amandeep, Ruhe and Standford 2004). These steps are summarized as follows:

Step 1. Characterize and understand: studying and classifying the characteristics of the project is performed at this step. This includes knowing the amount of resources available, number of people involved, project scope, quality criteria, etc.

Step 2. Problem definition: identifying the stakeholders and assessing their influence, and specifying feature requirements is performed in this step.

Step 3. Planning: examining multiple scenarios with variations on parameters such as effort (development effort, testing effort, etc.), company values, and risk, is

performed in this step. The results of this step is then communicated with the stakeholders.

Step 4. Execution: the plan is executed.

Step 5. Analyse experience: Data collected from Step 4 is analysed to appropriately direct further development on the software.

Step 6. Package experience and results: The analysis of the data from Step 5 is documented such that it can be used at a later point in time should a similar scenario is played.

Before the initial release, planners must examine the plans given by EVOLVE II's automated algorithm and find a balance between the information gathered in the early steps (1–3). These steps are iterative and gathering data (steps 4–6) after the initial release of a software could better inform the scope of a project in its later releases. This decision-making process can still be difficult for the planner because there are many variables that have to be considered, and the trade-offs between the alternatives are not necessarily derived easily from looking at the solution set as raw data.

1.1.2. ReleasePlanner™ [1]

ReleasePlanner™ is an online release planning tool that realizes the EVOLVE II method, providing planners with a solution set of optimal plans and possibilities. It algorithmically reduces a multitude of possible plans to a small set of optimal plans (Bhawnani and Ruhe 2005). In ReleasePlanner™, stakeholders vote on feature priority on a 1–9 scale where 1 indicates the least priority and 9 indicates the highest. Releasing features with high priority

[1] http://www.releaseplanner.com

votes sooner contributes to positive stakeholder satisfaction, while postponing or shifting those features back to a later release leads to stakeholder disappointment for a plan. Stakeholder satisfaction is indicated on a 5-point scale ranging from *very excited* to *very disappointed*, along with *surprised* and *very surprised*—neither of which are negative nor positive, but serve as markers for when an unexpected decision is made in a plan. *Surprise* occurs when a feature that has a low priority vote gets released earlier in the plan. Stakeholder feature points, or SHFP, are defined for each plan based on stakeholder satisfaction regarding features in conjunction with releases. ReleasePlanner™ also summarizes the degree of optimality for the plan. If a plan has 100% optimality, then there is no better plan possible in the sense of achieving a higher stakeholder feature points, under any given resource and technological constraints (Ruhe 2011).

Data from ReleasePlanner™ can be exported in spreadsheet format closely resembling the site layout. Important information relevant to release planning is included in this spreadsheet.

Requirement					Precedence & Coupling Constraints		Resource Consumption			
ID	Group	Requirement	Description	Pre-Assignment	Precedes	Coupled to	Budget	Design Effort	Development Effort	Testing Effort
1	Login	Account Creation	students or instructors are able to create an account		03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,		600	6	10	3
2	Login	Login	Students and instructors can log into their accounts		03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,		600	4	5	3
3	Exercises	List of Lessons + Navigation	List of lessons to be studied is presented, each lesson with a set of exercises.		04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,		1000	13	16	15
4	Exercises	Lesson Progress	Lesson progress (lessons accomplished / total number of lessons) is presented.				900	12	8	3
5	Exercises	List of Exercises	Each lesson features a set of exercises to be solved.		07, 08, 09, 10, 11, 12, 13,		900	5	10	3
6	Exercises	Exercise: Fill in the Blanks	A sentence is provided, in which the user fills in the information required to properly complete a sentence. This may include modifying words in brackets to the proper tense.		12, 13,		2500	25	30	20
7	Exercises	Exercise: Multiple Choice			12, 13,		1800	22	20	10
8	Exercises	Exercise: True or False			12, 13,		1600	18	20	10
9	Exercises	Exercise: Translation	Users are required to translate a one or more sentences.		12, 13,		3000	25	60	40
10	Exercises	Exercise: Dictation	Users listen to an audio and are required to type what they hear.		12, 13,		5000	30	60	20
11	Exercises	Grammar Summary	Users are presented with a summary of the grammar points required for the current exercise set.				1500	20	22	5
12	Exercises	Exercise Feedback	Users are able to go through their answers and review whether they answered correctly or incorrectly. A score is provided.				2000	30	17	15
13	Exercises	Exercise: Correction	Users are able to correct their exercises. Grade is adjusted accordingly.				1000	18	17	7
			Users can import a full chapter containing grammar							

Resources | Requirement | Stakeholders | Solution Set | Value | Alternative_Info | Excitement

Figure 1.1. Sample release planning data in a spreadsheet format. Note that several other spreadsheets are not shown in this figure. Data taken from ReleasePlanner™

1.1.3. Motivation

As previously stated, the decision-making process involved in choosing the most optimal plan for a software's release is not trivial because the inherent trade-offs between alternative plans are not easily observable. As such, my research interest lies in exploring answers to the following problems or challenges:

Problem 1. Planners can have different decision-making processes.

If we look at the human model of decision-making (Zeleny and Cochrane 1982) as a basis for decision-making in software release planning, different planners may have different methods and preferences regarding their decision-making process. As such, it is a challenge to support multiple types of decision-making processes. Moreover, decision-making in software release planning is made complex by multiple factors and constraints that are interrelated, leading to the Problem 2.

Problem 2. It can be difficult for planners to account for the interrelated factors of software release planning.

Meticulous examination of these factors is required to make a well-informed decision in software release planning. While it can be done through examinations of raw data in spreadsheets (see Figure 1.1).

Problem 3. It can be difficult for planners to compare alternative plans in order to be able to choose the best one.

This builds from the previous problem in that it involves comparing factors in several alternative plans rather than just a single one.

1.2. Approach

In order to find a solution to the previously stated research questions, my approach is to explore the use of visualization to support decision-making in release planning. Card, Mackinlay, and Shneiderman described visualizations as "the use of computer supported, interactive visual representations of data to amplify cognition" (Card, Mackinlay and Shneiderman 1999). They also provided several key ways that visualizations can amplify cognition, including: increasing memory and processing available resources, reducing information search, enhancing pattern recognition, encoding information in to a medium that can be manipulated. These cognitive benefits allow visualizations to function as frames of reference or temporary storages for human cognitive processes (Fekete, et al. 2008). Hence, visualizations are often used to augment human memory involving tasks that have a considerable cognitive load.

To further demonstrate the benefits visualizations can have for reducing cognitive load, visualizations have also been shown to be beneficial to managerial tasks. Lurie and Mason (Lurie and Mason 2007) compiled a number of visualizations and showed that many of them "speed up routine analysis tasks by making it easier to see correlations, outliers, and trends and to make comparisons." They speculated that visualizations may have managerial implications that includes efficiencies, cost reductions, improved productivity, new insights, increased information accessibility, and decision confidence—all of which outweigh the potential disadvantage of having a complex visualization that requires learning.

This thesis seeks to apply these benefits of visualizations to support the decision-making process in software release planning. It describes the design and implementation of a

strategic release planning support tool, STRATOS (STRATegic release planning Oversight Support). STRATOS is a hybrid visualization that visualizes potential plan outcomes and reveals the decision-making factors for several plans within a single view, making it possible to compare several plans at once. It visualizes data retrieved from ReleasePlanner™, thus complementing an industry-grade tool for release planning. Furthermore, it is designed to help planners identify patterns in the data, and make sense of the plans by reframing perspectives, promoting understanding, and communicating details of the data. My intention is to enhance the decision-making process by empowering different problem solving strategies and practices.

Overall, this thesis is concerned with ***how to support planners in choosing an optimal plan by visualizing the interrelated factors of software release planning.*** Hence, in the design and development of STRATOS, I concentrated on the following research questions, in order from the most straightforward to the least:

Research Question 1. How can a visualization be designed such that it helps planners see the trade-offs between plans at-a-glance?

Problem 3 implicates that for a visualization to effectively support decision-making in software release planning, it must simplify the planners' task of comparing alternative plans in a solution set. Hence, this thesis include the examination of existing visualization techniques and re-appropriating them to fit this need.

Research Question 2. How can a visualization be designed such that it visualizes the interrelatedness of the different factors of release planning?

Problem 2 implicates that for a visualization to effectively support decision-making, it must facilitate the planners' task of accounting for the interrelated factors of software release planning. Exploring how to provide easily identifiable visual elements and interactivity is therefore one of the focus of this thesis.

Research Question 3. How can a visualization be designed such that it supports multiple types of decision-making processes among different planners?

Problem 1 implicates that to effectively support decision-making in software release planning, visualizations must be able to support different types of decision-making processes.

1.3. Context and Scope

This thesis is mainly concerned with the exploration of the use of visualization in supporting the decision-making process involved in software release planning. Figure 1.2 illustrates the scope of my research within the intersection of the following domains of study: (1) Human–Computer Interaction (HCI) – which is concerned with the design and development of interaction between humans and technology. (2) Information Visualization (Infovis) – which is concerned with helping people understand data through the use of visualizations. Lastly, (3) Software Engineering Management –

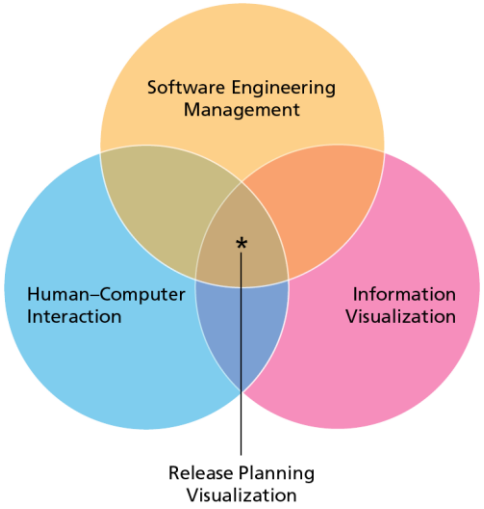


Figure 1.2. The scope of this thesis lies in the intersection of HCI, Infovis, and Software Engineering Management.

particularly in systematic management, which is concerned with ensuring that a software's market release is well-planned and executed.

1.4. Contributions

This thesis contributes the following:

Thesis Contribution 1. STRATOS, a hybrid visualization that visualizes potential plan outcomes and reveals the decision-making factors for several plans within a single view, making it possible to compare several plans at once.

Thesis Contribution 2. The qualitative evaluation methodology employed to study how planners used STRATOS and possibly similar visualizations

Thesis Contribution 3. The results of the study of STRATOS and its possible implications for other visualizations supporting decision-making in software release planning.

1.5. Research Acknowledgements

The work I have done for this thesis involves multiple collaboration with other researchers including fellow students, Tiffany Wun and David Ledo; David is a fellow master student who helped in the conception and early iterations of STRATOS, and Tiffany is an undergraduate research assistant I supervised who helped during the final iteration of the project and the qualitative study employed in this thesis. I also collaborated with release planning domain expert, Dr. Guenther Ruhe; and received much needed guidance from my supervisors, Dr. Anthony Tang and Dr. Sheelagh Carpendale. Nevertheless, I wrote this thesis as an account of my personal perspective over the collaborative work which I have

led. Henceforth, I am using the first-person singular pronoun—*I, my*—in reference to work done for the completion of this thesis.

1.6. Document Overview

In this Chapter, **Chapter 1 – Introduction,** I presented the background information for the motivation of this thesis, and the approach I employed in trying to support decision-making in software release planning. The remainder of this document is divided into four chapters with the following descriptions:

Chapter 2 – In Perspective. I present previous work in software development, release planning, and information visualization that puts my research in perspective. I also present the visualizations that inspired the design of STRATOS.

Chapter 3 – STRATOS: Design and Implementation. I elucidate the design guidelines of STRATOS, and then describe the hybrid visualization developed to instantiate these guidelines.

Chapter 4 – STRATOS: Study. I describe the qualitative study employed to examine how a visualization like STRATOS could support decision-making in release planning. I then present the results and elaborate on the implications it may have on the design of similar visualizations.

Chapter 5 – Conclusion. I summarize the contributions of my research and explore some avenues for future work.

Chapter 2

IN PERSPECTIVE

As stated in the previous chapter, visualizations have been used to support tasks that put considerable mental load on people (Fekete, et al. 2008). It is not surprising therefore, that there are many visualizations often used in software engineering. Moreover, previous research has been done on using visualizations to help in different aspects of software development and management.

In this chapter, I present several previous work in software development and information visualization that places my thesis in perspective. I describe standard visualization techniques that are currently used in software development and visualizations for software management, and note how my work builds on these. I then write about the visualizations I took inspiration from in creating STRATOS.

2.1. Visualizations for Software Development

At the inception of this thesis, I looked at how visualizations are currently being used in software development—starting from common methods that simply show software architecture to methods that show factors that affect how software development is managed. Examining these methods allowed me to conceptualize a hybrid visualization that supports the decision-making process of release planning, STRATOS, built from my understanding of the strengths and weaknesses of each.

2.1.1. Visualizations of Software Architecture

In software engineering, one practical use of visualization is to provide engineers with a standard way of visualizing a design or architecture of a software. The **Unified Modelling Language**, or UML (Rumbaugh, Jacobson and Booch 2004), is an umbrella of diagram drawing techniques commonly used by software engineers and developers to capture and portray requirements during the software development process. It provides them with constructs to build object-oriented models that are as close as possible to real-world models (France, et al. 1998). These constructs are visual components used in creating a variety of diagrams that show *structure* (such as **class diagrams**, Figure 2.1) and *behaviour* (such as **sequence diagrams**); thus, allowing software engineers and developers to see the software architecture and model user interaction.

The practicality of UML primarily lies on its simple yet effective visual depiction of software architecture. Furthermore, because it is standardized, software engineers and developers can use the diagrams to communicate ideas with one another. However, because

UML is specifically designed to visualize software architecture, it does not provide adequate constructs to visualize components of software management (resources, development scheduling, etc.). Hence, one must look for other methods in order to visualize software release planning factors.

While UML diagrams are inadequate for visualizing the factors of software release planning, it laid out one fundamental decision in the development of STRATOS—that is, the visualization must contain visual constructs that are close to real world models of release planning, making them identifiable to planners who have experience with UML.

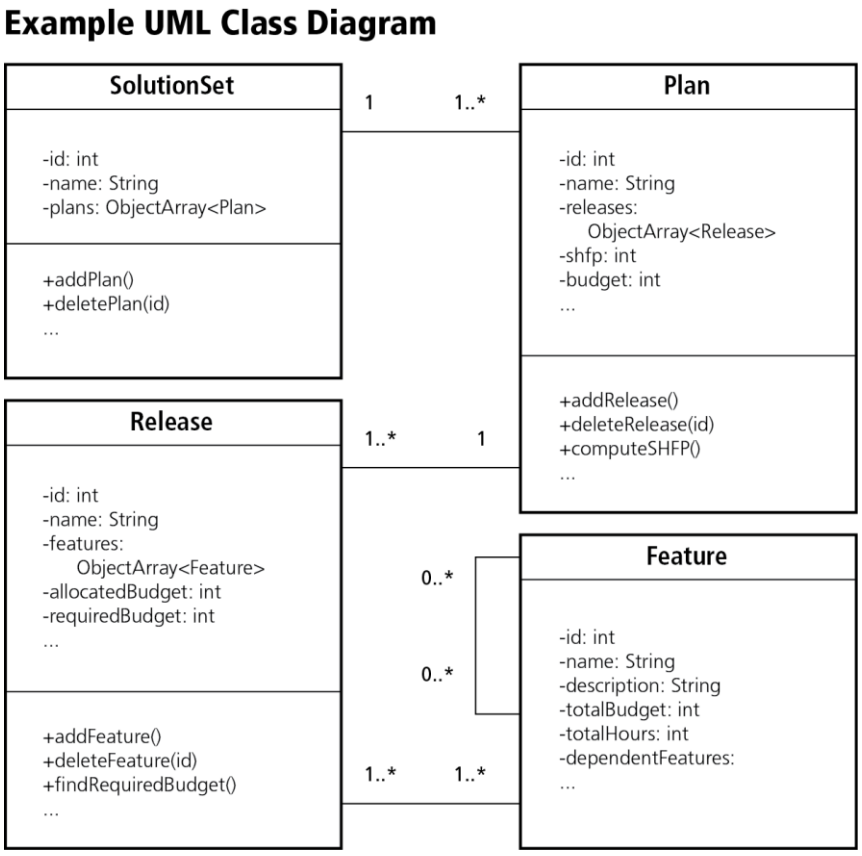


Figure 2.1. An example of a UML class diagram. Each *box* is a representation of a class—typically an *object*, as per object-oriented programming—divided into three parts: the class name (top), its attributes (middle), and its methods (bottom). It can also show relationships between classes, in this case for example, the SolutionSet class can have one or more Plans, but a Plan can only be in one SolutionSet (as portrayed by edges connecting the classes).

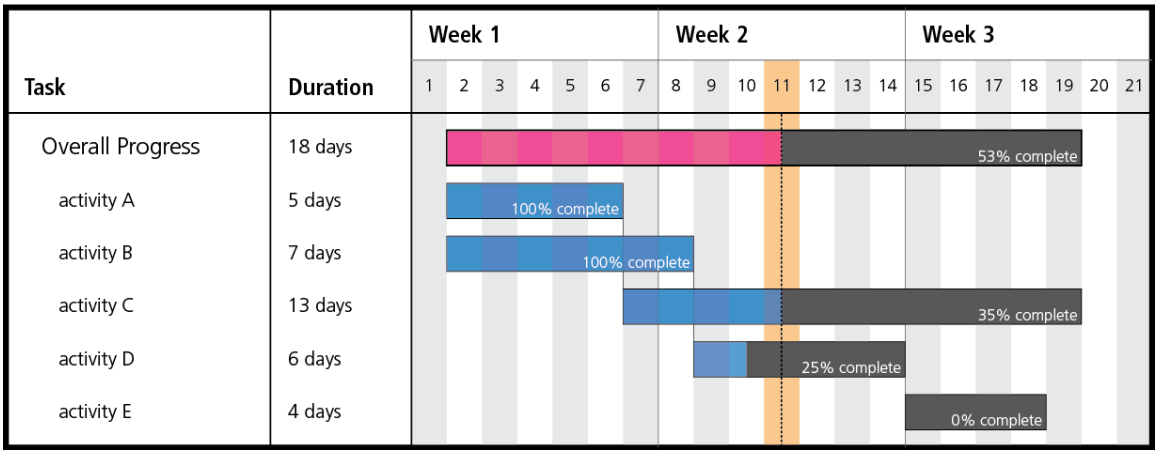


Figure 2.2. An example Gantt chart showing progress of work over a project. In this example chart, the current day is denoted by the yellow bar (day 11), the pink bar shows the overall progress made on the project, and the blue bars show progress on separate activities. The Project is broken down into five activities: A, B, C, D, and E, and shows that C is dependent on A's completion, D is dependent on B, and E on D. The chart also shows that progress on activity D is delayed.

2.1.2. Visualizations of Development Schedule

Aside from visualizing components of software architecture, visualizations are also extensively used in software engineering to visualize the development schedule for software. Unlike the visualization techniques I presented in Section 2.1.1, these visualizations show not only the software structure, but also the process of how it will be developed. One such visualization, the **Gantt chart** (Clark and Gantt 1923), allows software developers to visually chart the hierarchical break down of work over software components into different *activities* during development (see Figure 2.2). Thus, it is used for planning and directing the flow of personpower and time into activities and tracking work progress, effectively helping development teams in executing plans with minimal confusion.

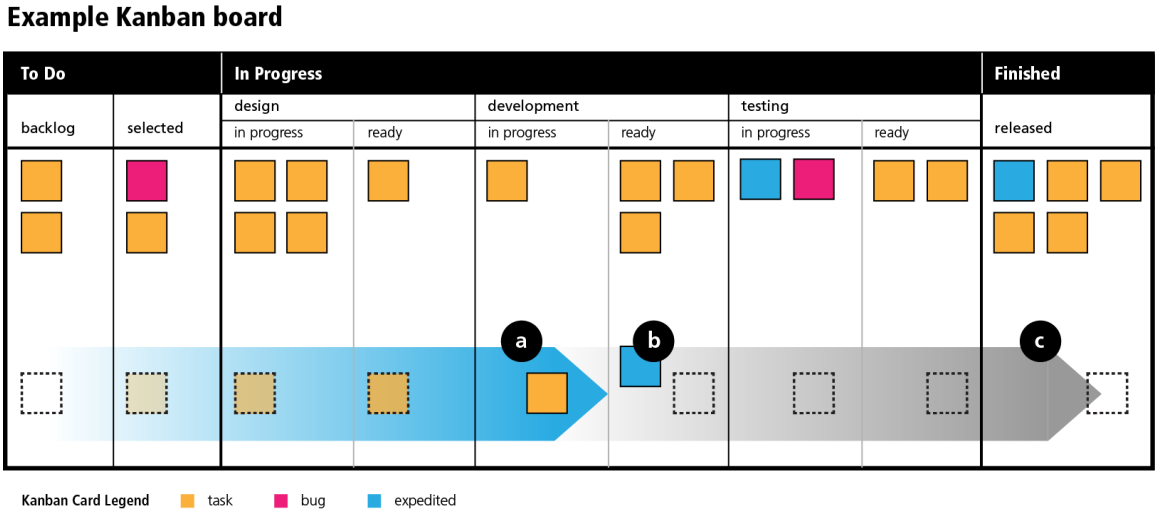


Figure 2.3. An example Kanban board. Each coloured square is a Kanban card deliverable (tasks, bugs, and expedited tasks). As illustrated at the bottom, the flow of a Kanban board typically flows from left to right: a deliverable starting at the *to do* pile is moved on to the *in progress* pile after a member of the development team starts working on it (a). After finishing the deliverable, it is then moved to the *finished* pile (c). Expedited tasks (b) take priority and can interrupt a task that is already in the development line.

Kanban (Anderson 2010), literally the Japanese word for *billboard*, is another method used in software development that visualizes the workflow of a development team. It depicts just-in-time development processes, where a feature is implemented only when there is an explicit customer request for it. As a consequence, development is represented as a large number of small *deliverables*. These deliverables are depicted with cards (called *Kanban cards*) that can be moved along a board signifying where it is in the development cycle. As seen on Figure 2.3, Kanban boards are typically broken down into *to do*, *in progress*, and *finished* work flow bins, while Kanban cards are selected from a backlog of deliverables. The cards are then moved along the board as they are designed, developed, and tested, until they are finished and released to market. It should also be noted that other development teams can change the details of the Kanban board according to their team's needs (e.g. they can add more details to the *in progress* bin such as *requirements gathering*).

This visualized workflow allows a team to track and analyse their progress to find ways to dynamically improve their schedule.

Both the Gantt chart and Kanban are effective methods of visualizing a team's development progress. If used correctly, both ensure the proper use of time. However, while both methods account for managing development time, they are inadequate for managing other resources (e.g. budget) and factors (e.g. stakeholder happiness) that are important in software release planning. This suggests that both rely on a pre-existing, well-thought out plan containing the right amount of money, time, and personpower for the development of prioritized features.

Putting these in perspective, I envision STRATOS to be a visualization tool that could complement these methods. This is because my research concentrates on supporting the decision-making process to come up with a plan before the development begins (and possibly when a change in plan is necessary). I focus on visualizing software release planning factors such as budget and development effort as opposed to focusing on the development progress.

2.1.3. Visualizations of Software Release Planning Factors

Release planning tools like ReleasePlanner™ provide basic visualizations such as bar and line graphs. While I am not trying to undermine their utility, these basic visualizations are typically focused on simple bivariate relationships. Typically, the complex, multivariate relationships inherent among the factors of software release planning, are not easily observed through these basic visualizations. The goal of introducing visualization to software release planning is therefore often the same: to increase the transparency of

solutions, showing why certain plans are suggested, and what the trade-offs look like between alternatives (Amandeep, Ruhe and Standford 2004).

Several authors have explored using different visual representations to support release planning; each visualizing specific factors in order to help planners make key decisions in managing the development or release of software.

2.1.3.1. Analysing Features

In software release planning, choosing which features should be implemented within a release is an integral part of its decision-making process. One way of doing this is to analyse whether a feature is truly integral to the success of the software even if the software's scope changes. As Wnuk et al. (Wnuk, Regnell and Karlsson 2008) stated in

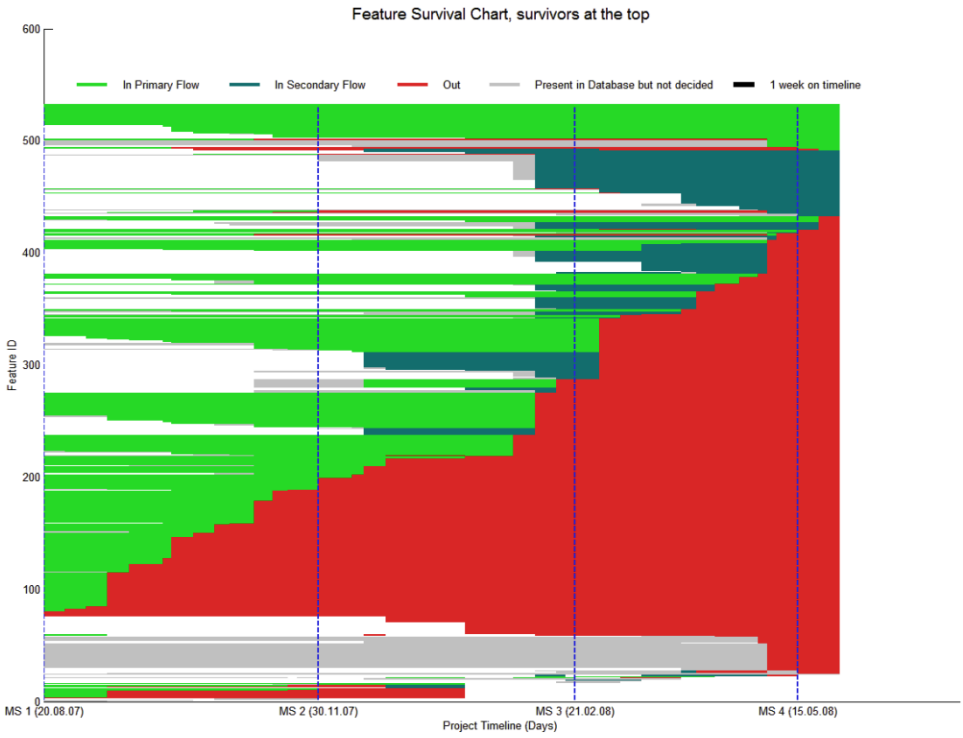


Figure 2.4. The Feature Survival Chart (FSC) reproduced from Wnuk, Regnell and Karlsson. © 2008 IEEE.

their research, development teams in industry are usually faced with scope changes from stakeholders and customers—when the scope of a software changes (typically after a *milestone*), development teams may find that previously planned features are no longer within the scope of the software. As such, it would be a waste of effort to still develop out-of-scope features for future releases. Hence, Wnuk et al. explored how feature life cycles can be represented in a two dimensional graph. Their goal was to help development teams in analysing requirements to find out whether or not certain features are still worth implementing for future releases of the software. They contributed two graphs: **Feature Survival Chart** (FSC) and **Feature Growth Chart** (FGC).

The feature survival chart (FSC) is a visualization of features and how the changes in the scope of the software affect them. Figure 2.4 shows an example of an FSC containing 531 features (Y-axis) across 9 months and four milestones (X-axis, M1–M4) when the scope of the software is updated. In this figure, the features are sorted by how long they remained in scope, with the ones on top being the longest *survivors*. Because the green lines depict features that remain in scope, proper placement of development effort can be seen whenever the graph appears greener at the most recent milestone (more features remained within scope). On the other hand, development effort could be considered as wasted should

the graph appear redder at the most recent milestone (more features are no longer within scope).

The feature growth chart (FGC), as seen on Figure 2.5, shows an overview of the full scope of the project. FGCs allow development teams to see trends as projects progress. In this example, the overall trend is that the number of out-of-scope features is increasing while the number of in-scope features is decreasing.

Wnuk et al. claimed that both FSCs and FGCs allow development teams to "construct valuable process efficiency measures" to improve requirements gathering and to avoid placing effort on developing features that will not survive a scope change. Furthermore, they have shown that both of these visualizations are useful for analysing the implications

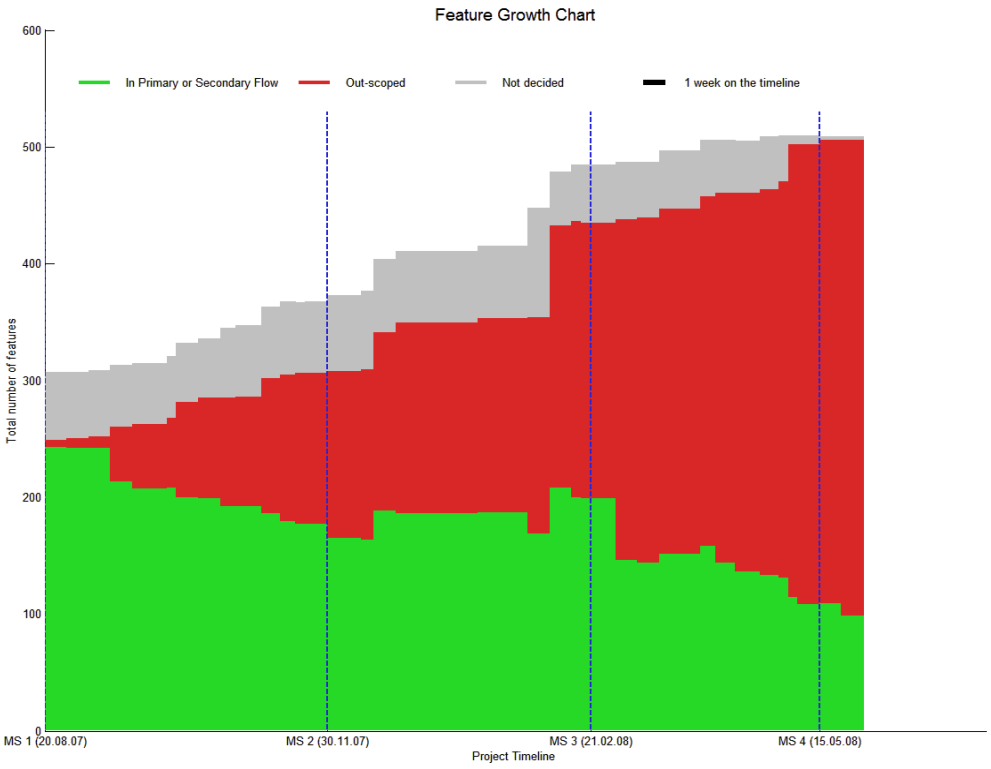


Figure 2.5. The Feature Growth Chart (FGC) reproduced from Wnuk, Regnell and Karlsson. © 2008 IEEE.

of decisions made on the software scope. However, because they both visualize data about the progress of features that are already implemented, they only show information on whether a reassessment of requirements is needed. They are not intended for predicting whether features will survive or become out-of-scope. Furthermore, because they focus on feature requirements, they only aid planners in deciding which features should be implemented. Much like the previously stated visualization methods, FSCs and FGCs do not support the visualization of other factors such as budget and development effort.

2.1.3.2. Analysing Requirements Interdependency

Carlshamre et al. (Carlshamre, et al. 2001), in their work on understanding the interdependencies of requirements in software release planning, illustrated a way to represent feature dependencies (see Figure 2.6). This included visualizing coupling (features that rely on each other), precedence (when a feature is required by another), cost, and value through a graph resembling a directed node-link graph. This is useful for showing functional dependencies for planning the course of development. However, this visualization does not account for the broader external factors that impact release planning (e.g. resource allocation and stakeholder preferences). This may be attributed to the fact that this research only used visualization for preliminary exploration, and the researchers admitted that further investigation is needed to examine the utility of this approach.

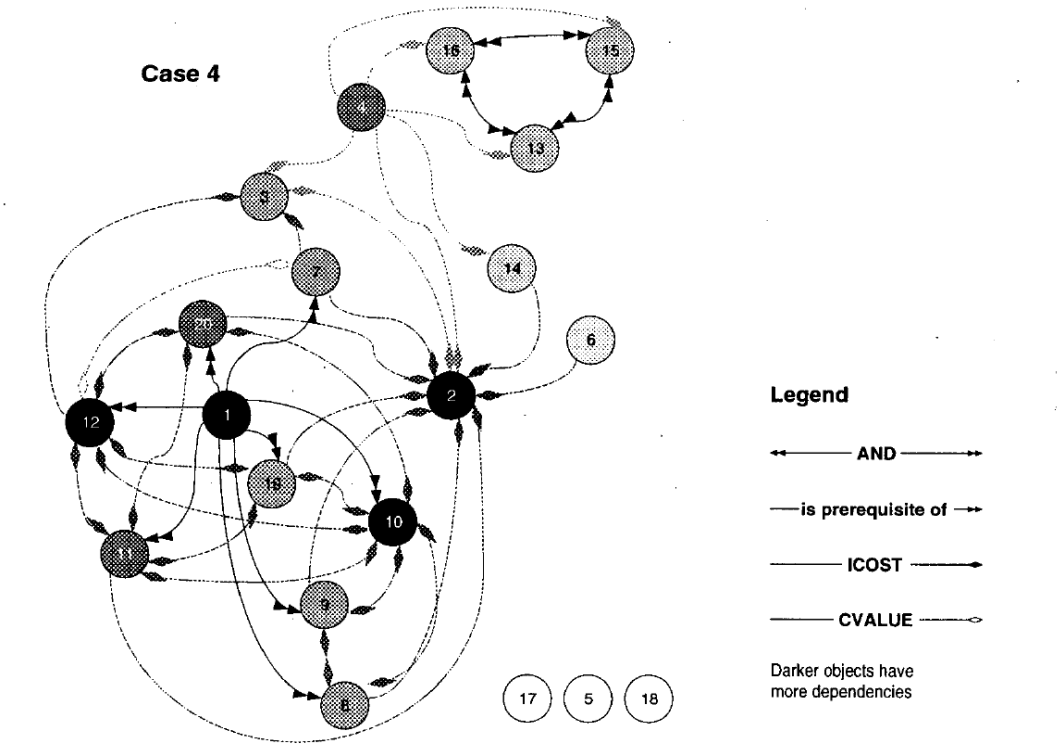


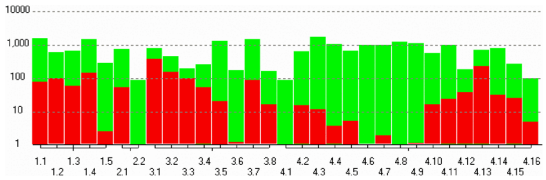
Figure 2.6. An example visualization of requirements interdependencies. Reproduced from Carlshamre et al. © 2001 IEEE.

2.1.3.3. Analysing Risks

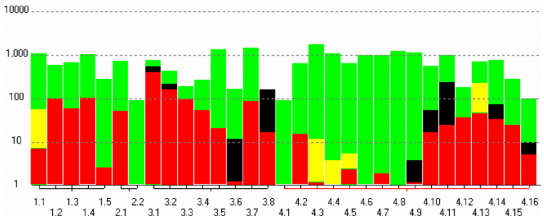
Feather et al. (Feather, et al. 2006) provided several representations that showed the requirements, risks, and risk options for the planning of a release. Their tool provides different representations and visualizations for: comparing risks, exploring the solution space as a trade-off between cost and benefit, and decision-making. Figure 2.7 shows a few of the basic, straight-forward visualizations that they used for each specific risk data set. These visualizations were described as self-contained and separate, with no mention of whether or not they are able to communicate with each other through Infovis techniques such as linking and brushing (Buja, et al. 1991). While the researchers claimed that each visualization was sufficient, switching between views to in order to accomplish multiple

tasks can be cumbersome. This is because view switching relies on people mentally integrating information across several views to find answers to their questions.

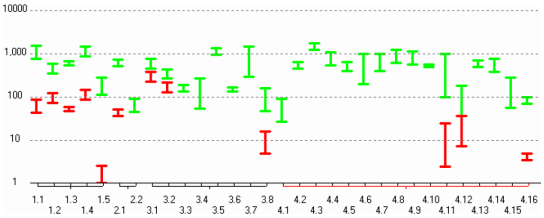
This prompted me to find other ways of presenting data that requires the least view switching, if none at all. Hence, in designing STRATOS, I began with the premise that several variables (the factors of software release planning) need to be visually accessible simultaneously to reduce the burden of view switching.



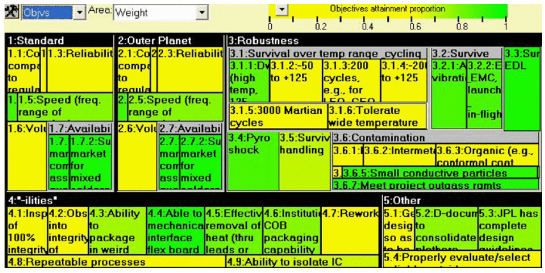
a. Bar chart of risk



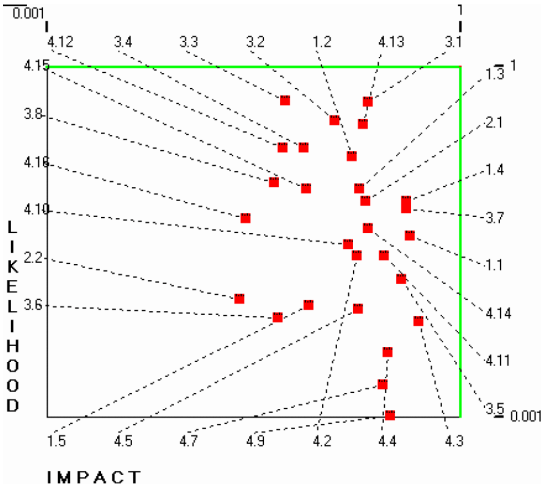
b. Bar chart comparison of risks



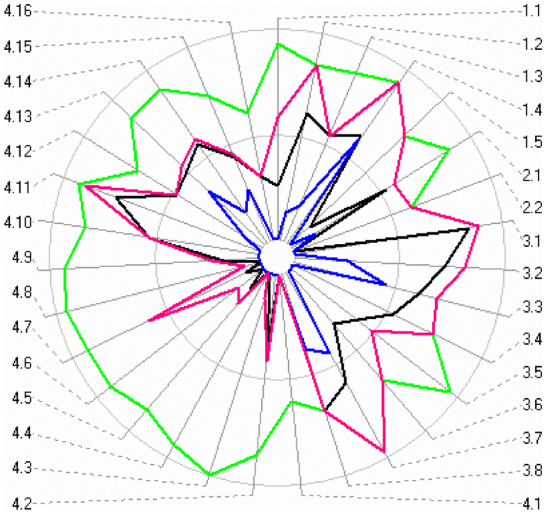
c. Range chart of risks



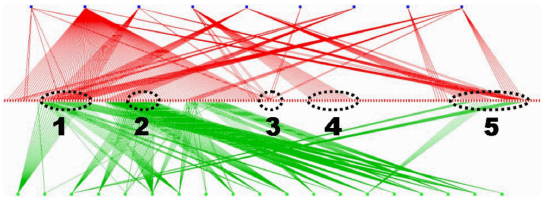
d. Treemap of requirements



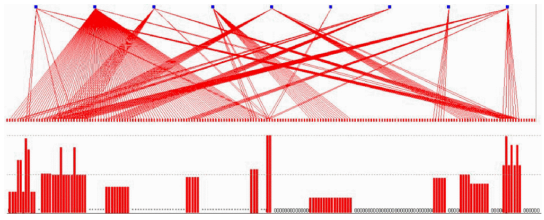
e. 2D chart of risks



f. Kiviat chart of design risks



g. Topology of needs



h. Topology of needs coupled with bar charts

Figure 2.7. Several visualizations showing risks assessment in software release planning. Each of these visualizations are separate views designed for specific tasks during risk assessment in release planning. Reproduced from Feather et al. © 2006 IEEE.

2.2. Visualizations that Influenced the Design of STRATOS

Based on what I have learned from the previously given visualizations, I examined pre-existing visualizations that could potentially show the interrelated factors of software release planning within a single layout. The approach I chose is similar to what Henry et al. (Henry, Fekete and McGuffin 2007) employed in the creation of NodeTrix—where they combined the advantages of two pre-existing visualizations into a hybrid visualization. Thus, I studied several visualization candidates to create STRATOS. The ones that were eventually used are described as follows.

The first visualization is the **Sankey diagram** (Sankey 1896) which was based on Charles Joseph Minard's drawing of Napoleon's Russian Campaign of 1812 (see Figure 2.8)—to which he claimed to promptly convey *"the relation not given quickly by numbers"* (Tufte 1983). Sankey diagrams have been used for depicting energy and material balances of complex production systems such as steam-engine production (Sankey 1896) (Schmidt

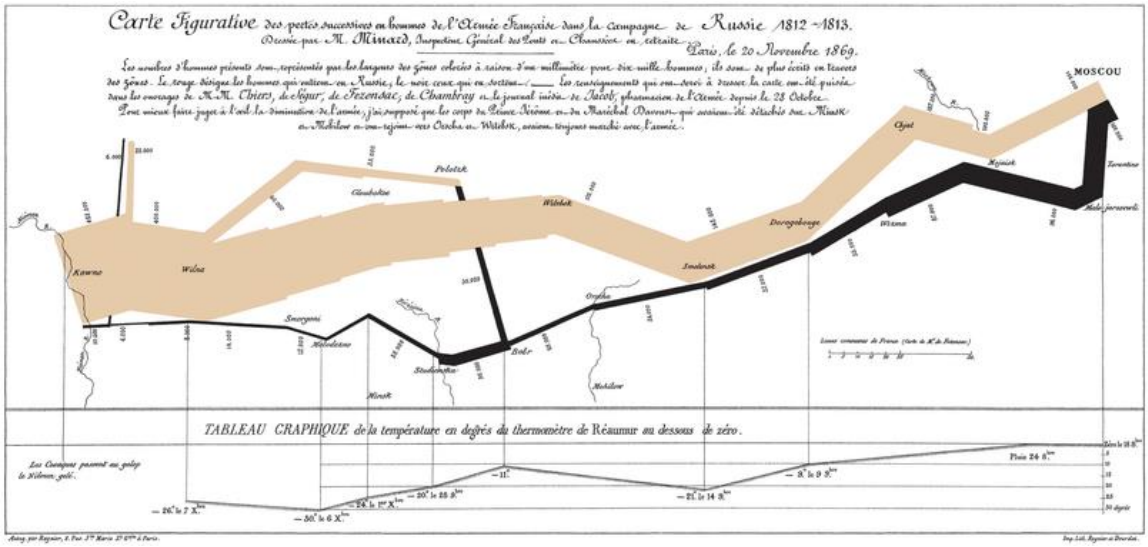


Figure 2.8. Charles Joseph Minard's diagram depicting Napoleon's Russian Campaign of 1812, which Edward Tufte considers to be "the best statistical graphic ever drawn" (Tufte 1983).

2008)—in which they were proven to be very useful in planning how to properly use finite resources. Reihmann et al. furthered this development by making Sankey diagrams interactive and useful for planning alternative flow scenarios (Reihmann, Hanfler and Froehlich 2005). This can be used to depict the flow of resource allocation within a plan—a release planning factor that is usually not depicted by previously mentioned visualizations. Thus, it is an appropriate candidate for the main visualization of STRATOS.

While the Sankey diagram is effective for depicting resource consumption, software release planning data has other properties that have led me to examine other visualizations as well. The second visualization I examined is **Parallel Coordinates** (Inselberg and Dimsdale 1990). Parallel Coordinates are graphical representations of multi-dimensional relations. Each axis of a parallel coordinate chart represents a dimension of data with two or more dimensions. For example, Figure 2.9 shows the eight planets of our solar system with the blue lines connecting them to their minimum, mean, and maximum surface temperatures.

One major aspect of the data I am concerned with is that it is highly comparative, containing information about several plans, releases, and features. Features, in particular,

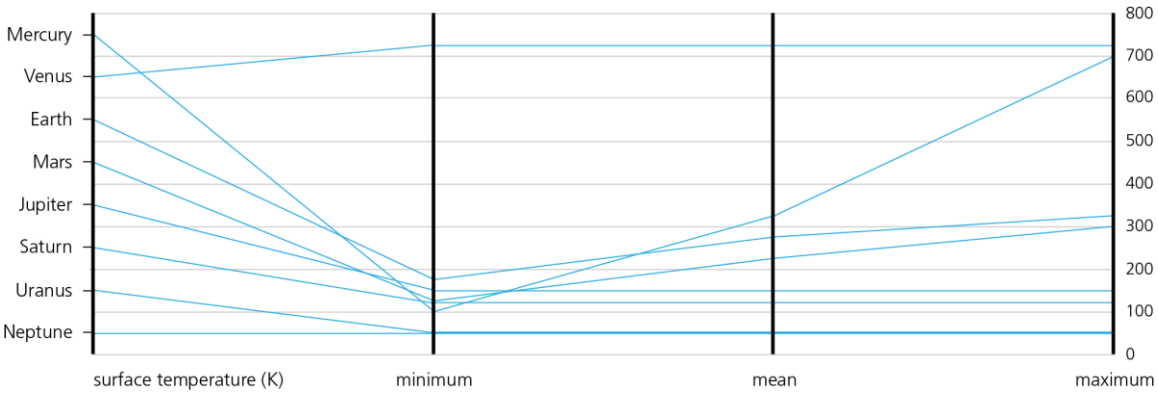


Figure 2.9. An example parallel coordinate chart showing the eight planets of our solar system and their minimum to maximum surface temperatures.

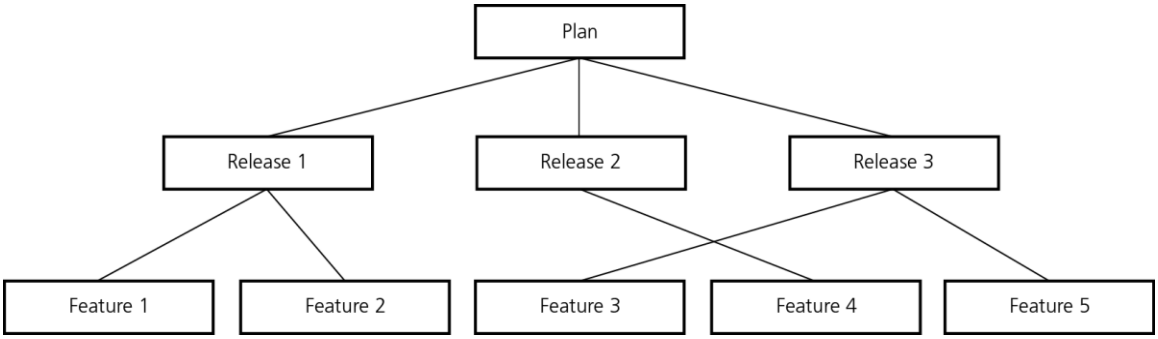


Figure 2.10. An example graph tree notation showing the hierarchy of a mock release plan.

remain constant among all of the plans in a solution set, with the difference being their priorities. This makes such data multivariate which can easily be visualized with parallel coordinates, making the visualization another appropriate candidate for the design of STRATOS.

The last visualization candidate is **graph tree notation** (Feiner 1988). It is a graphical layout that many people are familiar with—I chose the graph tree notation to visualize the hierarchical nature of software release planning data (see Figure 2.10).

These three visualizations—Sankey diagrams, parallel coordinates, and graph tree notions—served as main influential pieces of the hybrid visualization, STRATOS, developed to support decision-making in software release planning.

2.2.1. Parallel Sets

Since the development of the hybrid visualization used in STRATOS, a similar visualization technique called **parallel sets** (Kosara, Bendix and Hauser 2006) has since been brought up to my attention. Parallel sets, as seen in Figure 2.11 is a visualization which combines the pre-existing technique found in parallel sets and of displaying frequencies. The authors of this visualization specifically designed this visualization for the purpose of displaying

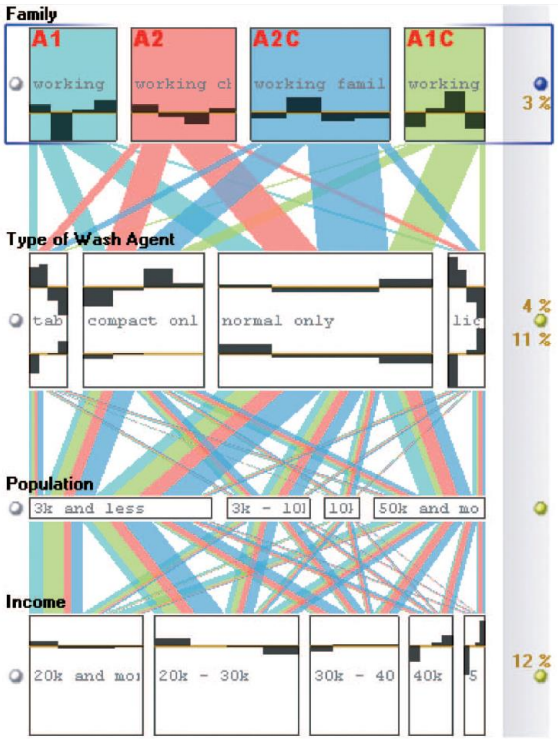


Figure 2.11. An example parallel sets showing the frequency distribution of families to different dimensions (type of detergent they use, their income, etc.) reproduced from Kosara et al. © 2006 IEEE.

categorical information, treating all of the dimensions as visually independent. Moreover, because this visualization focused on portraying categories, parallel sets have been modified to depict non-contiguous variables as its dimensions (with an option to show continuous dimensions).

While parallel sets bears a striking resemblance to the visualization approach presented in this thesis, there are some key differences. The visualization employed in STRATOS is designed with a more practical than theoretical purpose; that is, it is for the comparison of different plans in software release planning to support planners in choosing an optimal plan. As such, the dimensions used in STRATOS are alternatives of each other rather than different categories. Rather than splitting frequencies into different categories, the flow lines in STRATOS is split according to their allocation to the different releases and features that require them. I highlight more details of the differences between the visualization techniques used in parallel sets and STRATOS in this thesis' Chapter 3 Section 3.3.4. Nevertheless, parallel sets has been shown to help its users with identifying relationships in the data with fair ease. Because STRATOS' visualization is similar, this

arguably supports the idea that the visualization this thesis offers could help planners with identifying relationships in release planning data.

In the next chapter, I present STRATOS, describe its hybrid visualization and outline the design guidelines I employed in its design.

Chapter 3

STRATOS: DESIGN AND IMPLEMENTATION

In this chapter, I highlight the process I underwent in designing STRATOS to provide planners with decision-making support during software release planning. I outline the seven design guidelines I followed to ensure that the visualization provides decision-making support. I then present the end result, a hybrid visualization technique that combines the flow visualization of Sankey diagrams and the multivariate visualization of parallel coordinates within a tree layout. I describe its visual representation and the interaction techniques it employs. Lastly, I explain the method with which the visualization is drawn algorithmically.

The design process also involved frequent and iterative feedback from a release planning expert, Dr. Guenther Ruhe[2]; and it is owing to this collaboration that I am able to meet the requirements of supporting decision-making in software release planning.

[2] http://ruhe.cpsc.ucalgary.ca

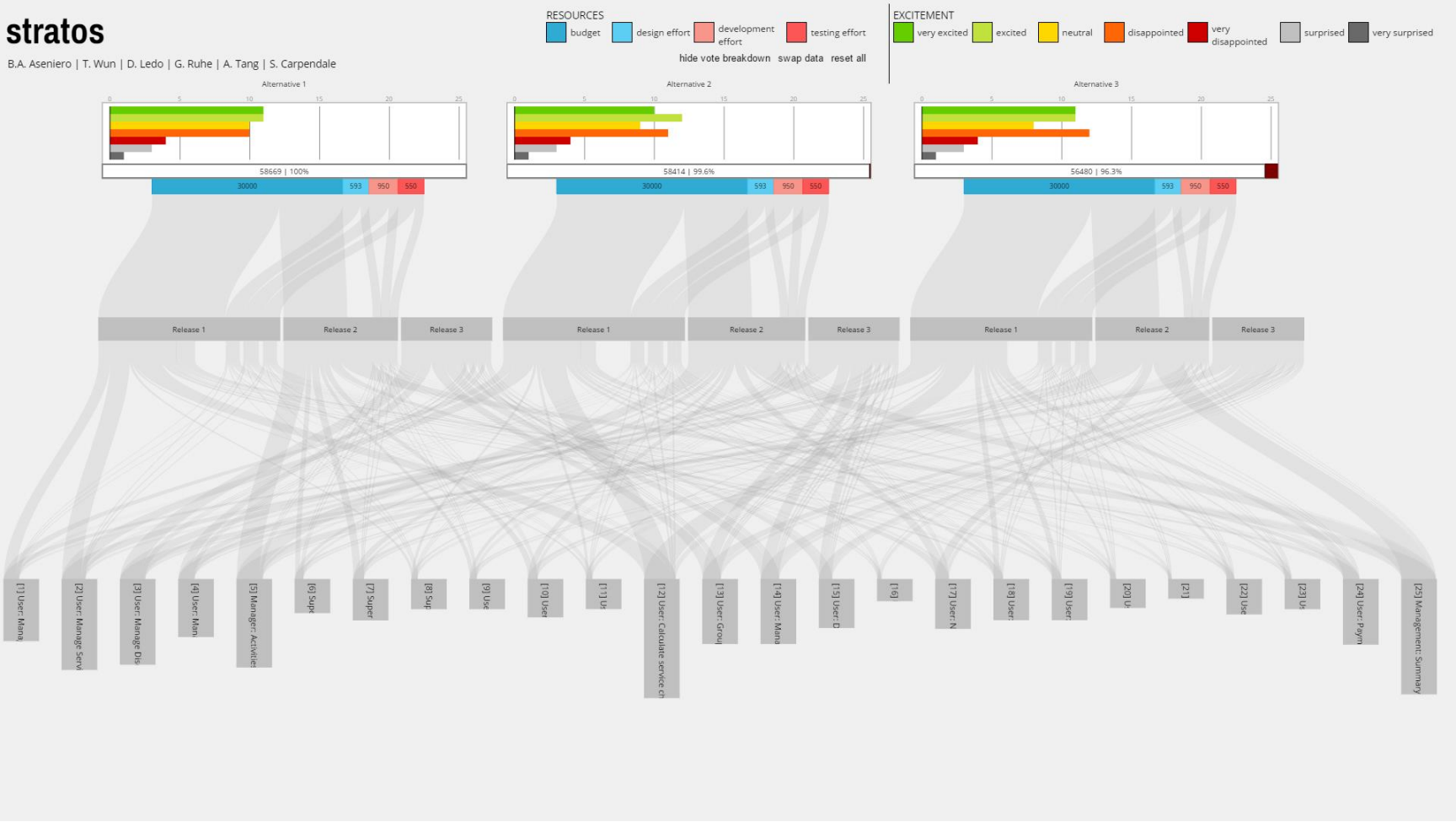


Figure 3.1. The hybrid visualization used in STRATOS. The visualization shows a solution set from ReleasePlanner™ containing several alternative plans to choose from within a single, unified layout, and does not require view switching.

3.1. STRATOS: Definition

As stated in the introduction (Chapter 1, Section 1.2), STRATOS—whose name comes from a portmanteau of Strategic software release planning Oversight Support—is a visualization tool designed to support the decision-making process involved in software release planning. Complementing ReleasePlanner™, STRATOS visualizes the important factors of release planning within a single, unified layout (see Figure 3.1). This is to ensure that all of the relevant factors are available to the planner at-a-glance. Furthermore, Stratos was implemented with interactive brushing (Buja, et al. 1991), allowing every component to interactively reveal relationships within the data.

3.2. Design Process

The main methodology used in developing STRATOS' is a *design study methodology* (Sedlmair, Meyer and Munzner 2012). This method follows a framework of nine stages within three top-level categories (shown in Figure 3.2) ensuring that one gets the most out of their collaboration with the domain experts whom one is collaborating. I chose to apply

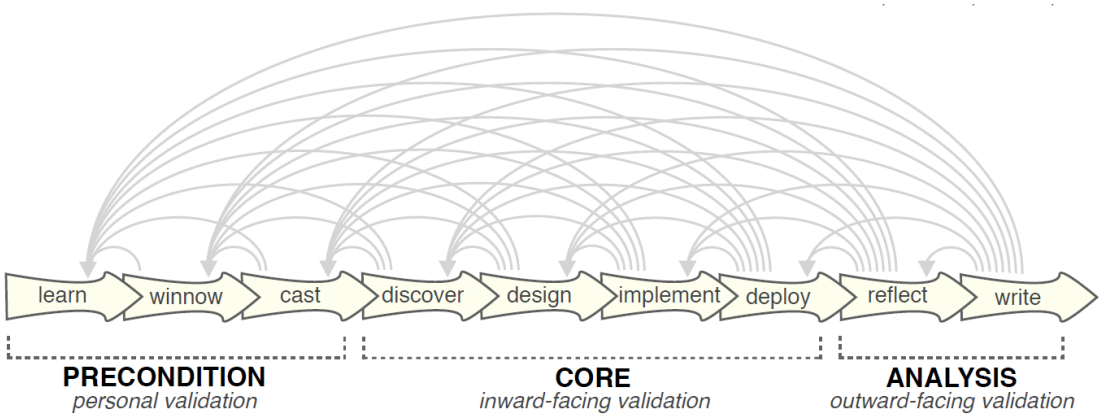


Figure 3.2. The nine stages of design study methodology categorized into three top-level categories. Reproduced from Sedlmair, Meyer, and Munzner © 2012 IEEE.

this methodology in developing STRATOS to gain sufficient knowledge of software release planning practices and the requirements for supporting the needs of my target end-users (planners).

Precondition. This level of the framework contains the stages in which visualization designers learn about the topic of the work or process they hope to support, winnow (or carefully select) possible collaborators, and cast collaboration roles with the domain experts on the topic. Hence, during these stages of the design study, I went over a review of the literature, finding ones that placed my thesis in perspective (as presented in Chapter 2). I worked closely with a domain expert on software release planning (both in practice and research), Dr. Ruhe, who also spearheaded the development of the release planning tool, ReleasePlanner™, that provides the data visualized by STRATOS.

Core. This level of the framework contains the stages in which visualization designers discover the challenges and problems they need to overcome, design the abstraction of data, and implement a solution. Hence, during these phases, I asked the help of the domain expert to give considerable insight into the field of software release planning and the decision-making process that takes place. He helped identify important patterns and relationships between the factors of software release planning that are not immediately evident—providing design guidance for STRATOS. This prompted me to create a design that specifically highlights these relationships and patterns which are not easily seen with current traditional tools for release planning or basic visualizations. It is also during these stages in which I sought to connect my research questions (Section 1.2) to my design goals. For example, the first question *"how can we design visualizations that support multiple types of decision-making approaches among different planners?"* (Research Question 1)

is based on my assumption that planners are individuals with different approaches to decision-making. This assumption is based on a model of human decision-making in which there are two basic approaches: the *outcome-oriented* approach and the *process-oriented* approach (Zeleny and Cochrane 1982). Decisions from an outcome-oriented approach are based on the predicted outcome, seeking answers to *what* or *when* questions. On the other hand, decisions from a process-oriented approach are based on the understanding of *how* a good result can be achieved. This knowledge has driven the design of STRATOS to consider supporting both approaches.

Analysis. This level of the framework concludes the design collaboration through reflection. During the analysis phase, I performed a qualitative evaluation of STRATOS involving participants with knowledge of software release planning. I studied and reflected upon the ways they interacted with the visualization—noting key observations that, to a certain extent, allowed me to validate the utility of STRATOS and its design guidelines.

The remaining sections of this chapter provide more details about the design and implementation stages. The deployment and reflection stages are discussed in Chapter 4.

3.2.1. Design Guidelines

Recall that my thesis explores ***how to support planners in choosing an optimal plan by visualizing the interrelated factors of software release planning***. Hence, seven design guidelines were developed to be followed in the design of STRATOS. These guidelines are the end result of brainstorming and design sessions with other researchers in HCI and Infovis—taking guidance from the requirements and other information gathered during discussions with the domain expert. Furthermore, the development of these guidelines

included consideration of the related literature and processes presented in Chapter 2, and were improved by the reflecting upon the results the qualitative evaluation of STRATOS. Arguably, these guidelines have a potential to be useful in designing future similar visualizations that aim to support decision-making in software release planning.

The underlying design goals are as follows:

Design Guideline 1. Consider as many as possible factors.

Knowing that the conditions of multiple factors of software release planning is important for planners to be able to make good and well-informed decisions, the visualization design must take into account visualizing as many factors as possible.

Design Guideline 2. Provide a holistic view.

Visualizations for supporting decision-making in software release planning should not only be able to show the factors but must also be able to show how they relate to one another. A holistic view allows decision makers to consider most of the factors with considerable ease rather than trying to do so while switching between views.

Design Guideline 3. Support comparison among alternative plans.

Comparing trade-offs among possible alternative plans is at the heart of decision-making in software release planning. Therefore, plans must be shown as distinct visual elements within the visualization to help planners easily identify them as alternatives to one another. At the same time, consistency across representations should be employed such that they could be visually compared. At-a-glance comparison of alternative plans could effectively enable this comparison through

the use of visual variable that emphasize the major differences among alternative plans.

Design Guideline 4. Support multiple decision-making strategies.

Different planners often have different approaches on deciding what the best alternative plan is in regards to their project's goal. An interactive visualization should allow planners to explore the data according to their own preferences; by letting them find possible outcomes (outcome-oriented approach) and or by helping them better understand a given solution (process-oriented approach).

Design Guideline 5. Support details-on-demand (Shneiderman, The Eyes Have it: A Task by Data Type Taxonomy for Information Visualizations 1996)**.**

While visually conveying information allows planners to do simple comparisons at-a-glance, they should still be able to access detailed information such as the numeric values of the visualized data. This could help planners to accurately distil information that look similar when visualized.

Design Guideline 6. Minimize required interactions.

Minimizing interaction over-head by avoiding excessive clicking, selecting, etc., while still providing full visualization and data access will make interacting with the visualization more pleasant. This could lead to better acceptance of the tool, making it easier to be integrated with other support tools or methods that the planners may already using.

Design Guideline 7. Support individual and collaborative exploration of the data.

Release planners may explore alternative plans individually or in a group, such as when having a meeting. Hence, there is an advantage to allow planners—either

individually or as a group—to explore the visualization simultaneously according to their own practices and as a communicative tool. This could be possible by designing the visualization to run over a large screen display for many people to see. Another way of doing this is to provide awareness between planners who are not collocated (e.g. creating a web-based application that allows the exchange of information between multiple clients).

In summary, Design Guideline 3 call for the use of distinct but identifiable visual elements to allow comparison of alternative plans at-a-glance, in addition, Design Guideline 6 calls for minimizing the interactions required for comparing plans. Thus, these guidelines provide a solution to Research Question 1. Design Guideline 2, as well as 1 and 5, concentrates on providing a holistic view of the factors paired with details-on-demand. Careful choosing of visual elements and layouts enables the visualization to depict the interrelatedness of the factors of software release planning; thus, providing a solution to Research Question 2. Lastly, Design Guideline 4, as well as 1 and 5–7, concentrates on what aspects of the data should be visualized and how interactions should be supported. By taking as many factors into consideration, and allowing for interaction to begin anywhere the planner wishes to, these guidelines potentially enable multiple decision-making strategies and afford a *freedom-of-choice*. As such, they are means of finding a solution to Research Question 3.

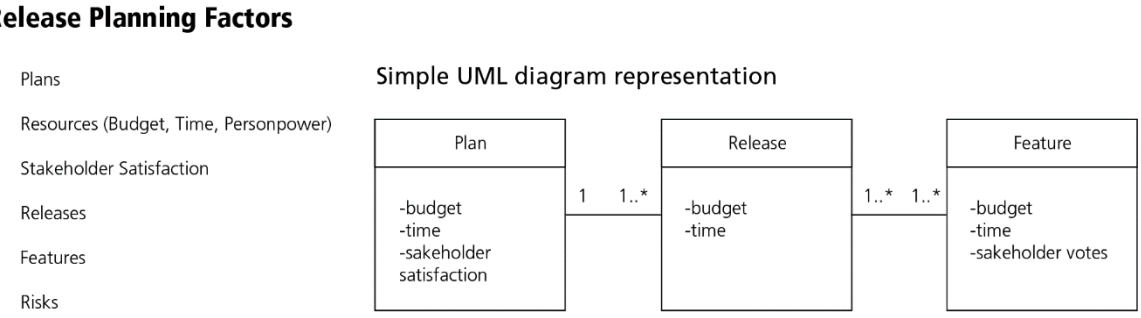


Figure 3.3. A UML diagram representation of the structure of the basic release planning factors.

3.2.2. Creating the Hybrid Visualization

As stated earlier, I followed the guidelines mentioned in the previous subsection to design STRATOS as a tool for decision support. Most importantly, some of the guidelines (specifically design guidelines 1–3) helped dictate how the abstraction of data should be done. The approach I used to provide a holistic view (Design Guideline 2) is through a hybrid visualization that brings together several aspects of existing visualization techniques. As seen on Figure 3.3, I turned to the technique used in UML class diagrams (Chapter 2, Section 2.1.1) to choose how the different factors of release planning should be abstracted (Design Guideline 1). Based on studying the data from ReleasePlanner™, my knowledge of software release planning, and advice from the domain expert, I chose *plans, releases,* and *features* as the main visual elements of my visualization, with other factors such as *resources* and *stakeholder satisfaction* distributed among them. Assessment of *risk factor* has been left out as a matter of scope, but in principle, it can be integrated into the visualization as well.

Further examination of the data reveals that certain visualizations are best suited to represent them. Figure 3.4 on the next page shows an overview of my observations about

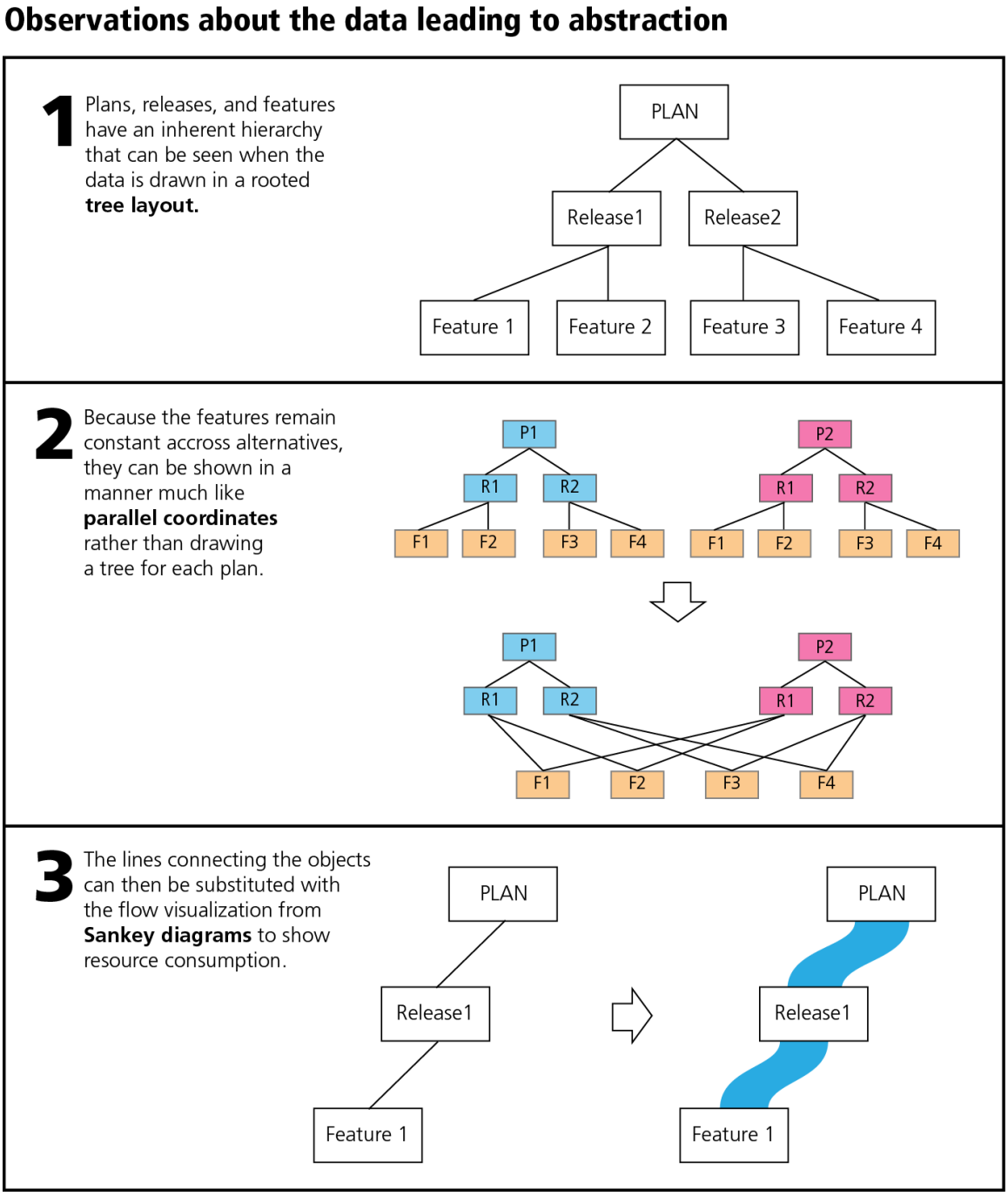


Figure 3.4. Some major observations about the data which shows the basis for STRATOS' design.

software release planning data which provided the basis for how I combined existing visual representations to create the hybrid representation of STRATOS.

In the next section, I describe the visual representation of STRATOS in detail.

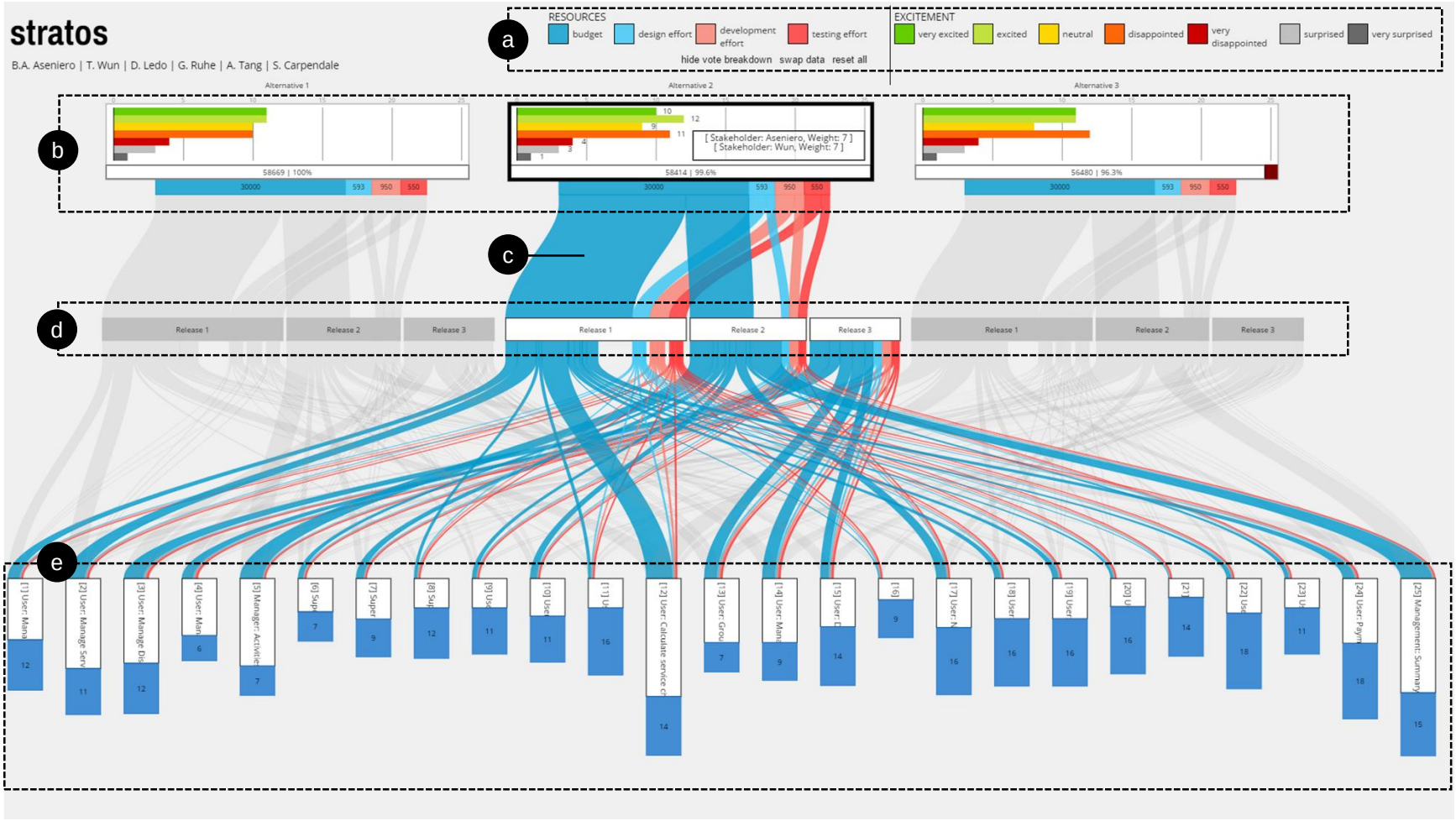


Figure 3.5. A view of STRATOS in which the middle alternative plan is highlighted. (a) Legend for the colour representations of resources and excitement levels. (b) The boxes representing the alternative plans within the solution set. (c) The flow diagram visualizing the flow of resources into (d) the alternative plan's releases, and eventually to (e) the features.

3.3. Visual Representation

As previously mentioned, STRATOS is a hybrid visualization that integrates Sankey diagrams (Sankey 1896) and parallel coordinates (Inselberg and Dimsdale 1990) in a forest or multiple tree view (Feiner 1988). Figure 3.5 shows an overview of STRATOS. Starting at the top right hand side (Figure 3.5.a), there are two legends: one for the set of colours representing the resources and another for those representing the excitement levels of stakeholder satisfaction.

3.3.1. Plans

Each plan depicted in STRATOS can be thought of as a hierarchy containing resource consumption, releases, and features. The overall view is a small *forest* with one *tree* representing each alternative plan. The hierarchy shows plan headers at the top (Figure 3.5.b), releases in the middle (Figure 3.5.d), and the set of features at the bottom (Figure 3.5.e). Since all alternative plans contain the same set of features—though they have been given different priorities—the trees representing the alternative plans also share the same set of features. This sharing of the same set of features can visually suggest that the plans are alternatives for the same software.

For each alternative plan, a header containing a bar chart representing stakeholder satisfaction is depicted at the top of the hierarchy. As shown in detail in Figure 3.6.a, the bar chart is composed of seven bars corresponding to each level of excitement from *very excited* to *very disappointed*, including *surprised* and *very surprised* (for more information on these excitement levels, see Chapter 1, Section 1.1.1). These excitement levels are further summarized with the stakeholder feature points to an overall degree of optimality

for the plan. This is represented by the white bar located just beneath the bar chart (Figure 3.6.b).

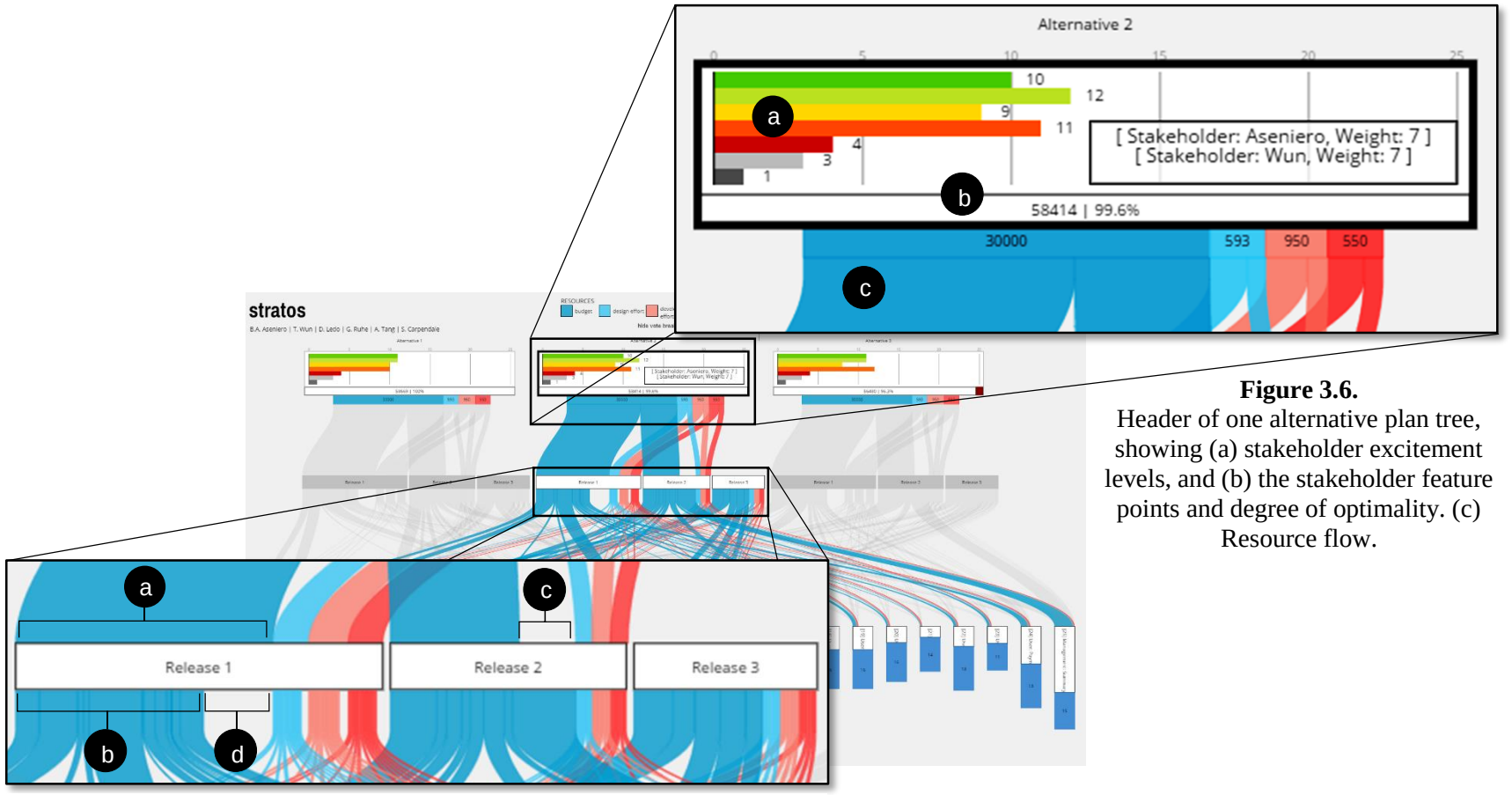


Figure 3.6.
Header of one alternative plan tree, showing (a) stakeholder excitement levels, and (b) the stakeholder feature points and degree of optimality. (c) Resource flow.

Figure 3.7. The flow diagram shows the allocation of resources as (a) the initial allocated amount of resources, (b) the actual amount needed (for budget). (c) Gaps in the incoming re-sources mean that the release needs more of that resource, while (d) gaps in the outgoing resources mean it needs less of that resource to implement the features within it.

In Figure 3.6.c, the initial amount of available resources is shown (blue: budget, light blue: design effort, pink: development effort, and red: testing effort). The flow represents the resource allocation among plans—perhaps one of the more crucial factors considered in decision-making—with the thickness of the flow mapped to the available (or required) amount of the resource it represents. The flow of resources shown from the plan to the releases and from the releases to the features (Figure 3.5.c) function similarly to parallel coordinates where plans, releases, and features are the axes.

3.3.2. Releases

In the data visualized in Figure 3.5 (and shown in detail on Figure 3.7), the middle plan contains three releases: Release 1, containing the set of features to be released at first launch of the software; Release 2, containing the set of features to be released at a later time, through a patched update; and Release 3, containing the remaining subset of features which are postponed due to resource constraints. Hence, as seen on Figure 3.7, Release 3 does not receive any incoming resources. Each release is represented with a horizontal bar labelled with the release's number with its width corresponding to the total amount of resources needed to implement all of the features included in the release.

Figure 3.7.a shows the flow of resources into the three releases within an alternative plan. The flow visualization flowing into the release shows the amount of resources allocated for the release, while the flow visualization flowing out of the release shows the actual amount of resources required to implement the features in that development cycle (see Figure 3.7). Here, planners can see discrepancies between the planned resource allocation and the actual required resources (see Figure 3.7.c and Figure 3.7.d).

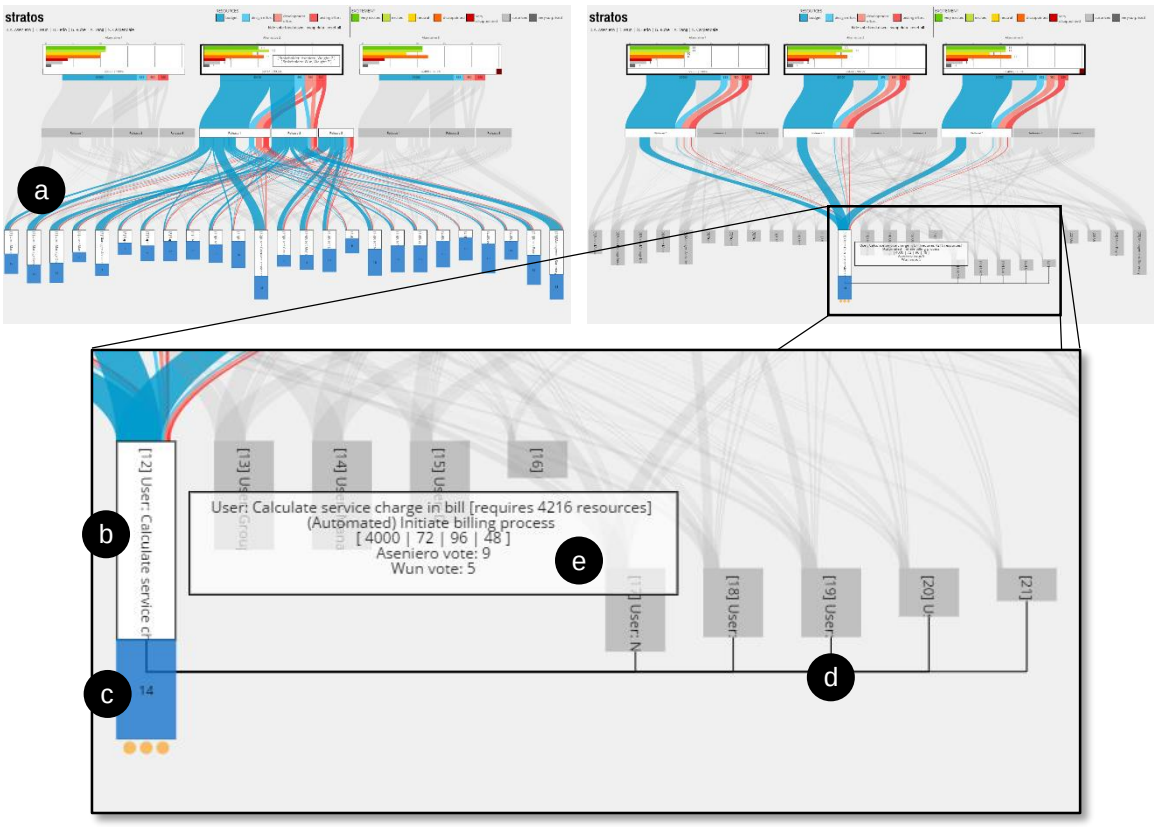


Figure 3.8. (a) Features are represented by a stacked bar graph at the bottom of the visualization. The stack shows (b) the amount of resources the feature requires and (c) its stakeholder votes regarding its priority. Clicking on the feature shows (d) dependent features. (e) Tooltip displaying detailed information about the feature including the breakdown of its stakeholder votes.

3.3.3. Features

At the lower part of visualization, the features of the software are listed (shown in Figure 3.8.a). This list of features works as a stacked bar chart where each feature is represented by a stacked bar. The height of the white bar (Figure 3.8.b) represents the amount of resources the feature requires (i.e. the longer it is, the more resources the feature needs), while the height of the blue bar (Figure 3.8.c) represents the consolidated stakeholder votes on the priority of the feature. The breakdown of these votes per stakeholder can be seen on the tooltip of the feature (Figure 3.8.e).

As stated earlier in Section 3.2.2, the factor of risk has been left out of the scope of STRATOS' visualization. However, as stated in Chapter 1 Section 1.1.1, stakeholders in ReleasePlanner™ also vote on their perceived risk factor of a feature, as such, this risk value could also be included in this representation of a feature by adding in another bar to the stack representing the risk factor. The overall risk of the plan can be added as category to each of the plan header (much like the representation of the stakeholder feature points).

In its current iteration, the ordering of the features through the x-axis is based on an ascending order of feature IDs. Other logical ordering of the features could be considered. For example, *sorting* methods (either ascending or descending) based on their required resources or on their stakeholder priority votes, and *binning* methods such as clustering the features closer to the releases they get implemented in to lower the number of overlapping flow lines.

When comparing a plan or a release, amber dots appear at the bottom of some features' stack (as seen on Figure 3.8) signifying the number of plans or releases it belongs to among those that are being compared. For example, when two releases are highlighted and compared, one amber dot under a feature means that the feature belongs to only one of the highlighted releases, while two dots means it is in both.

3.3.4. Comparison with Parallel Sets

As stated in Chapter 2 Section 2.2.1, the visual representation presented above bears a striking resemblance to parallel sets. As noted earlier, parallel sets is similar to STRATOS in that they both take the idea behind parallel sets and extend on it. The key differences between the visualization techniques are outlined below:

Key Difference 1. Depiction of discrepancies in the flow of resources.

STRATOS, as described in Section 3.3.2, allows the planner to examine discrepancies regarding the allocation of resources in a release. As such, the visualization was purposefully designed to add gaps between the different flow lines should a discrepancy in the flow occur, something that parallel sets do not account for because it accounts for frequencies.

Key Difference 2. The tree layout of STRATOS.

STRATOS, in aiming to provide a distinct but identifiable visual representation for plans, has been designed to emphasize the hierarchical nature of the data, and automatically places the releases under the header of the alternative plan they belong to. This can arguably be reproduced by manipulating the dimension axes of parallel sets, however it is not done so automatically.

Key Difference 3. The feature representation of STRATOS.

As shown in Section 3.3.3, the feature visual element encodes a value on its height. In parallel sets, the height of the box representing a dimension do not encode any value in particular.

3.4. Interaction

Through interaction, the planner can have access to all the details of the data. STRATOS was designed such that a planner can begin interacting anywhere in the visualization. This gives the planners freedom and flexibility to appropriate the tool to their own approach to decision-making (Design Guideline 4). For example, they could begin examining the

features first (bottom–top approach), or start examining the stakeholders' excitement level first (top–bottom approach). Interacting on each visual element shows a tooltip that provides more details about the data it represents thus supporting details-on-demand (Design Guideline 5). Furthermore, interacting directly with the visual elements was implemented, eliminating the need for menus and other similar techniques (Design Guideline 6). For example, to find the features scheduled to be implemented within a certain release, a planner simply needs to brush over the release rather than having to go through a menu and selecting a *show features* command. To describe the different interactions a planner can do while using STRATOS, this section goes over each interaction from a top to bottom approach regarding the different parts of the visualization. To facilitate ease of use when using STRATOS in a group setting (Design Guideline 7), it was intended to be usable on large screen displays. As such, the interactions described in this thesis are based off touchscreen interfaces (e.g. SMARTboard); hence, basic touch interactions such as *tapping* and *pressing and holding* are mentioned. On a traditional desktop environment however, the former maps to *clicking*, while the latter maps to *hovering.* In addition, STRATOS was implemented as a web application such that it can be scaled to provide support of non-collocated collaborations.

At the top of the visualization, tapping an alternative plan highlights the flow of resources, releases, and features related to that alternative plan (as seen on Figure 3.5), with the middle alternative being highlighted). Tapping again deselects the alternative plan, removing its highlight. Pressing and holding (for less than a second) on an alternative plan shows a tooltip containing the stakeholders and their corresponding weights.

The same interaction also applies to the releases in the middle of the visualization (Figure 3.5.d). In this case, the highlighted elements will be all the features that belong to that release, the header of the alternative plan the release belongs to, and the flow of resources coming in and out of the release (shown in Figure 3.9). Planners can highlight multiple releases from different alternative plans at once to compare them (e.g. they could look at the differences of two releases regarding the features they implement). Pressing and holding shows a tooltip containing the numeric amount of resources the release requires and the amount of resources allocated to it (Figure 3.9.c). Tapping on a release puts it in focus until it is tapped again.

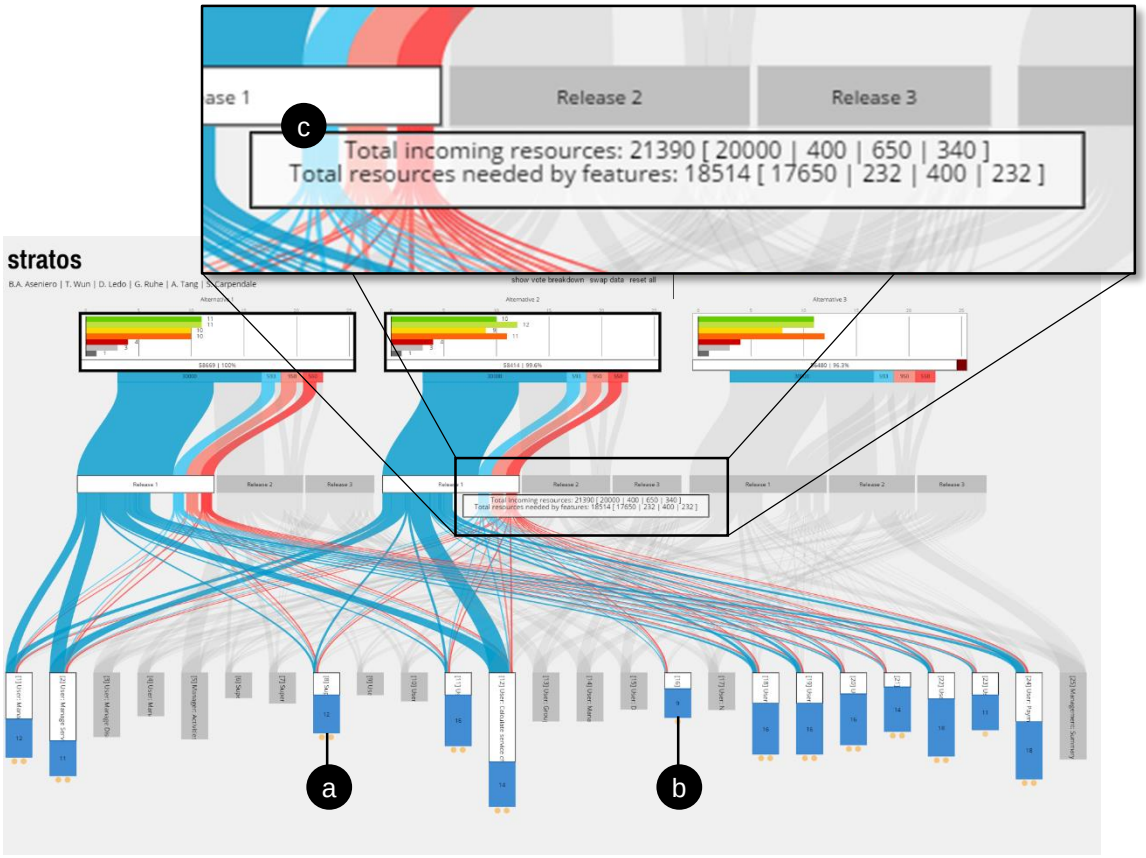


Figure 3.9. Selecting the same release number within two alternative plans allows a planner to visually compare them. In this example, both Release 1 of Alternatives 1 and 2 are selected, highlighting their resource allocations and the features included in both (a) or just one (b). (c) Tooltip containing detailed information about the release.

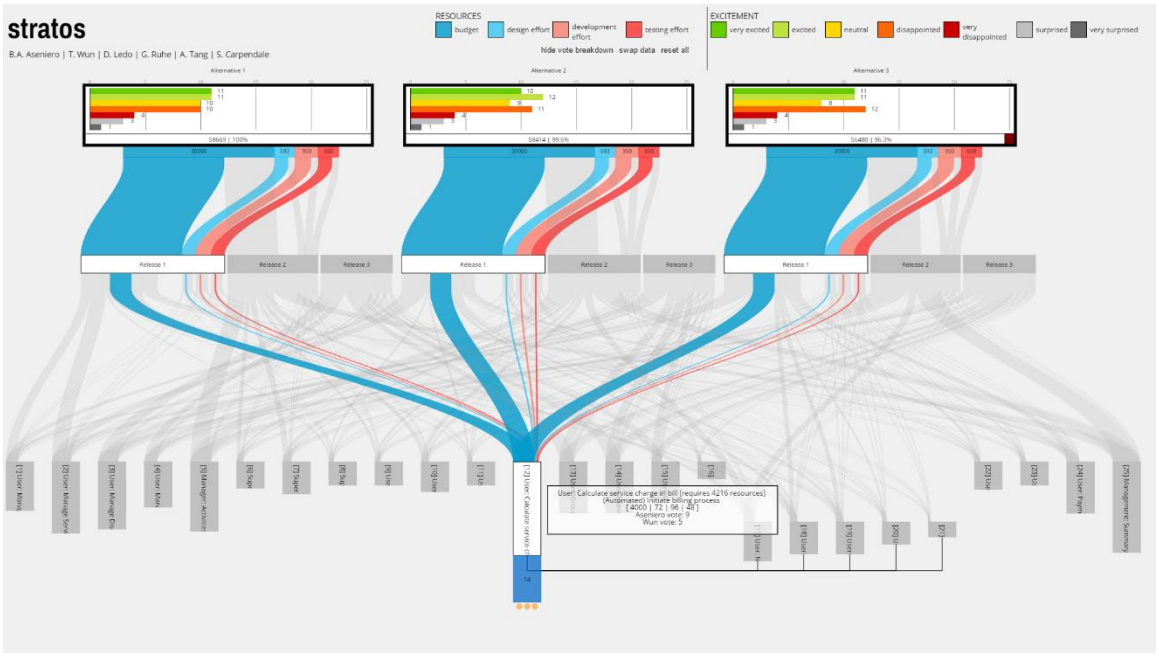


Figure 3.10. Shows the state of the visualization when a feature is selected by the planner.

Features are highlighted from the bottom–up. As shown in Figure 3.10, tapping on any of the features highlights all of the releases to which the selected feature belongs to within all alternative plans. It also highlights the stacked bar graph representing the amount of resources it requires and the stakeholder votes it received. Pressing and holding on the feature brings up the tooltip containing details about the feature (details about this tooltip is discussed in section 3.3.3 and shown in Figure 3.8.e). Tapping on a feature reveals its dependent features should they exist, and—as with alternative plans and releases—puts it in focus until it is tapped again. The dependent features will be moved slightly downwards and be connected by a line linking them to the feature upon which they depend (shown in Figure 3.10).

Other interaction methods are also implemented to improve STRATOS' usability. First, resources can be filtered out by choosing the resource type on the legend. All resource

types that are marked with an *x* on the legend will not be highlighted during interaction with the visualization (shown in Figure 3.11). This can be used by planners whenever they wish to focus on a specific type of resource(s). Second, a *reset all* button that clears all highlighted elements is available should the visualization become cluttered with a number highlighted elements. This is useful whenever a planner wishes to rapidly clear all highlighting rather than tapping every highlighted element to toggle off existing highlights in order to re-examine the data.

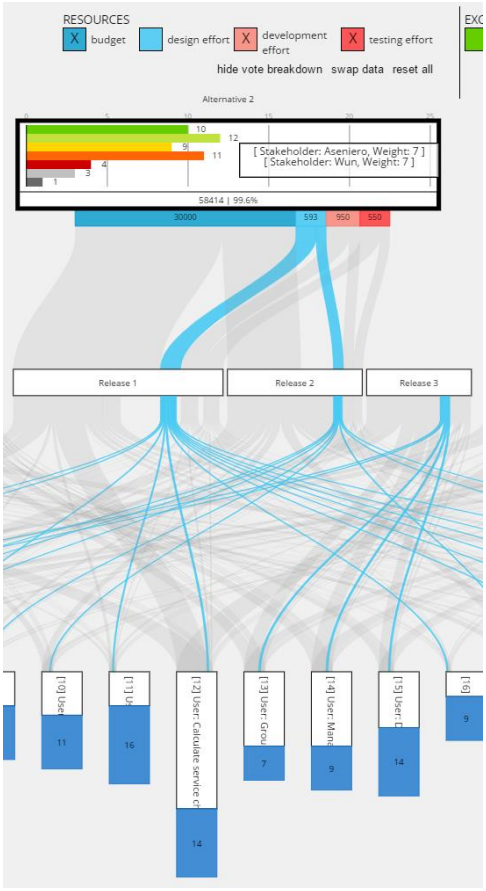


Figure 3.11. Filtering resource types.

3.5. Implementation

STRATOS was implemented as a web application written in HTML5 and JavaScript. The data it visualizes comes from the spreadsheet data generated by ReleasePlanner™.

3.5.1. Drawing Algorithm

In order to draw the visualization described in section 3.3, I used a combination of basic algorithms. In this subsection, I go over the basic drawing algorithms I used to draw each individual visual elements in STRATOS. I describe the algorithm I used to generate the visualization as a whole. While this algorithm performed well for use in the qualitative

evaluation of the visualization, it is yet to be optimized. All algorithms here after are written in pseudocode.

Some important data structures that I left undefined but are used in the algorithms are *Points* and *Lists* and operations such as *drawBezierCurve* and *drawRect*. These pertain to data structures and operations that are usually included or have counterparts within basic programming languages. It should be assumed for example that a Point structure contains *x* and *y* coordinate values and operations such as *translateX* and *translateY* whose respective parameters move the point in 2D space. The operations such as *drawBezierCurve* and *drawRect* also pertains to basic drawing operations in programming languages. They can also be thought of as mathematical functions that draw a Bezier curve and a rectangle.

```
1.  Flow(pointA, pointB, width){
1.
2.      // points
3.      Point a = pointA
4.      Point b = pointB
5.      Point c = pointA.translateX(width)
6.      Point d = pointB.translateX(width)
2.
7.      // control points
8.      Point Ca = pointA.translateY(CTRL_PT_CONST_A)
9.      Point Cb = pointB.translateY(CTRL_PT_CONST_B)
10.     Point Cc = pointC.translateY(CTRL_PT_CONST_C)
11.     Point Cd = pointD.translateY(CTRL_PT_CONST_D)
3.
12.     // Use basic path drawing methods
13.     draw(){
14.             BeginPath()
15.             drawBezierCurve(a, Ca, Cb, b)
16.             drawLine(b, d)
17.             drawBezierCurve(d, Cd, Cb, c)
18.             drawLine(c, a)
19.             EndPath()
20.     }
21. }
4.
```

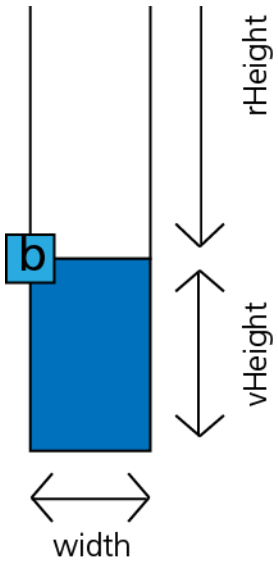
Code Listing 1. Pseudocode snippet of how to draw a single flow visual element.

3.5.1.1. Flow Diagrams

Flow diagrams in STRATOS are composed of a *starting point*, an *ending point*, and the thickness (*width*) of the flow. As seen in Code Listing 1 lines 13–20, the flow diagrams are drawn using Bezier curves (Piegl and Tiller 1995). To draw a flow diagram, a path with four points (*a*, *b*, *c*, and *d*) are needed. Points *a* and *b* are given as input parameters; point *a* being the first point at top of the flow and point *b* being the first point at the bottom. Points *c* and *d* are calculated by translating points *a* and *b* along the *x*-axis using the width of the flow. This width corresponds to the value of the resource it represents. Control points are calculated based on a constant and the distance between the top point and the bottom point. For example, the control point *Ca* is found by translating point *a* to 35% the distance from point *a* to *b* along the *y*-axis. This can be seen in Figure 3.12 which illustrates how a flow diagram is drawn.

```
1. Feature(origin, width){
2.      // Note:
3.      // The Feature data-structure also contains other information
4.      // about the feature such as name, dependent features, etc.,
5.      // and other methods
6.
7.      Point origin = origin
8.      int width = width
9.      int rHeight = toPixels(total_amount_resources) // amount of resource required
10.     Int vHeight = toPixels(stakeholder_votes) // amount of stakeholder votes
11.
12.     // Features are drawn using a basic rectangle drawing method
13.     draw(){
14.             drawRect(origin, width, rHeight)
15.             drawRect(origin.translateY(rHeight), width, vHeight)
16.     }
17.
18.     drawFlows(){ ... }
19. }
```

Code Listing 2. Pseudocode snippet of how to draw a single Feature visual element.



calculated from translating point *a* along the y-axis using the *rHeight*.

3.5.1.2. Features

The basic components needed to draw a feature visual element (see Figure 3.13) are its *origin point* (the feature's top left corner), *width*, the *height (rHeight)* corresponding to the amount of resources it needs, and the *height (vHeight)* corresponding to the amount of stakeholder votes it received. While it is not shown in Code Listing 2, the Feature data structure also includes other information such as the feature ID, name, description, list of dependent features, etc.

A basic rectangle drawing method is needed to draw the stacked bar graph representing a feature. As seen in Code Listing 2 lines 13–16, the first rectangle is drawn from the origin using a basic rectangle drawing method that takes in a point, a width, and a height—in this case, the *rHeight*—as parameters. The next part of the stack is drawn using the same

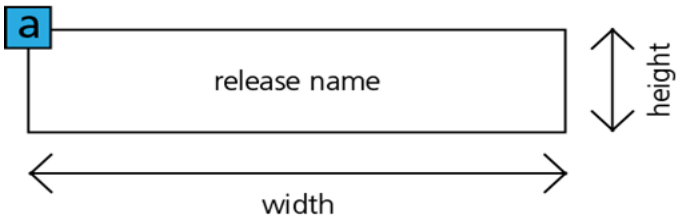


Figure 3.14. Shows a release visual element drawn from an *origin (point a), width,* and *height.*

method, but using *vHeight* as the new height and a new origin point calculated by translating the original origin point along the y-axis using the *rHeight.* This ensures that the new rectangle is drawn after the previous rectangle.

```
1. Release(origin, width, height){
2.      // Note:
3.      // The Release data-structure also contains other information
4.      // about the release such as name, list of features, etc.,
5.      // and other methods
6.
7.      Point origin = origin
8.      int width = width
9.      int height = height    // amount of resource required
10.
11.     // Releases are drawn using a basic rectangle drawing method
12.     draw(){
13.             drawRect(origin, width, height)
14.     }
15.
16.     drawFlows(){ ... }
17. }
```

Code Listing 3. Pseudocode snippet of how to draw a single Release visual element.

3.5.1.3. Releases

Much like features, the basic components needed to draw the visual element for releases include an *origin point*, a *width* corresponding to the amount of resources allocated for the release, and a *height* (see Figure 3.14). Again, a basic rectangle drawing method is used to draw the release visual element as shown in Code Listing 3.

3.5.1.4. Plan

Drawing the visual elements for a plan requires a few basic drawings and a combination of the previously mentioned visual elements. This is because the visual element for a plan is

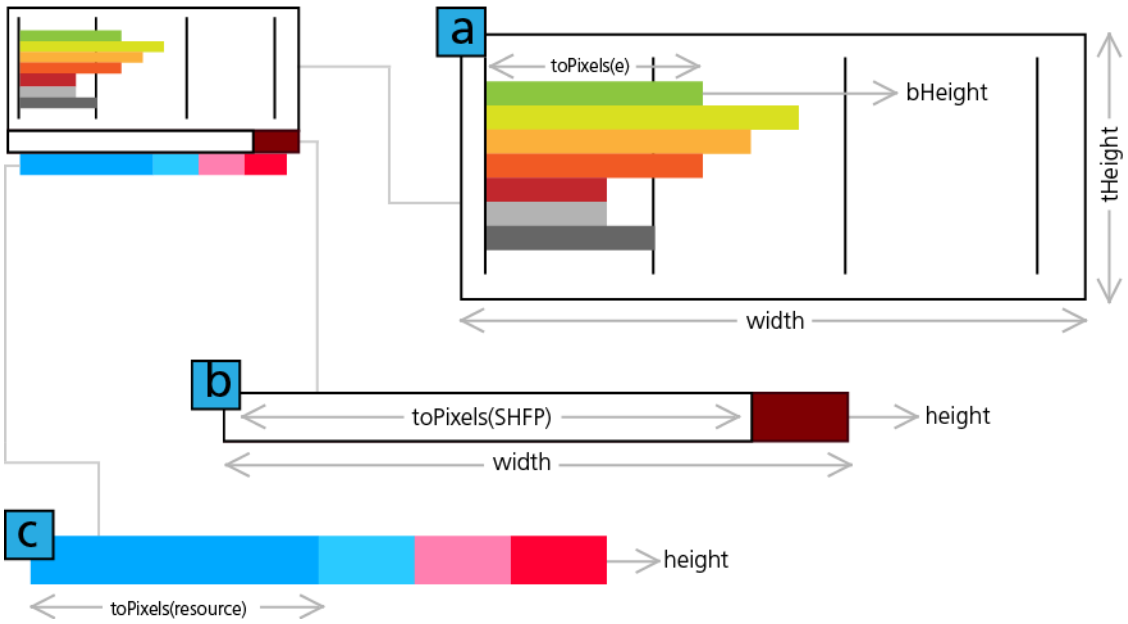


Figure 3.15. The breakdown of the top portion of an alternative plan into three components. (a) The stakeholder satisfaction bar chart, where the width of the individual bars is calculated from the data (denoted by the toPixels() method), (b) The stakeholder satisfaction points bar, and (c) the initial allocation of resources.

the tree hierarchy composed of the stakeholder satisfaction bar chart, releases, features, and the flow of resources in between the hierarchies. To draw the plan, first, the top portion (plan header) of the hierarchy is drawn. This includes the stakeholder satisfaction bar chart, the stakeholder feature points bar, and the initial resource allocation bars. Code Listing 4 outlines the methods used to draw this top portion of the plan hierarchy and is illustrated in Figure 3.15.

The basic components needed to draw the stakeholder satisfaction includes an *origin point*, *width*, *total height (tHeight)* corresponding to the height of the rectangle that will contain the bar chart, and *bar height (bHeight)* corresponding to the height of each individual bars within the bar chart. As seen on Code Listing 4 lines 12–20, the outer rectangle is drawn first, followed by the individual bar graphs whose width values come from the excitement

levels data. These data values are translated into pixel values by the *toPixels()* method. To draw the stakeholder feature points bar, the components needed include a new *origin point*, *width*, and *height.* Both this width and height correspond only to the outer boundaries of the bar, while the width of the bar representing the stakeholder feature points is calculated

```
1.  Plan(){
2.      // Note:
3.      // The Plan data-structure also contains other information
4.      // about the release such as name, list of releases, etc.,
5.      // and other methods
6.
7.      int SHFP
8.      List resources
9.      List excitementLevels
10.
11.      // Draws the stakeholder statisfaction bar chart
12.      drawSatisfactionBarChart(origin, width, tHeight, bHeight){
13.             drawRect(origin, width, tHeight)
14.             for (i from 0 to excitementLevels.length){
15.             fillColor = excitementLevels[i].color
16.             value = toPixels(excitementLevels[i].amount)
17.             drawRect(origin.translateY(i * bHeight), value,
18.                     bHeight)
19.             }
20.     }
21.
22.     // Draws the degree of optimality/SHFP bar
23.     drawSHFPBar(origin, width, height){
24.             fillColor = red
25.             drawRect(origin, width, height)
26.             fillColor = white
27.             drawRect(origin, toPixels(SHFP), height)
28.     }
29.
30.     // Draws the initial allocation of resources
31.     drawResourcesBar(origin, width, height){
32.             prevWidth = 0  // previous width
33.             for (i from 0 to the number of resources){
34.                     fillColor = resources[i].color
35.                     value = toPixels(resources[i].amount)
36.                     drawRect(origin.translateX(prevWidth), value,
37.                             height)
38.                     prevWidth += value
39.             }
40.     }
41.
42.     // draws the header of the plan by drawing all of its components
43.     drawPlanHeader(origin, width){
44.             drawSatisfactionBarChart(origin, width, T_HEIGHT_CONST,
45.                     B_HEIGHT_CONST)
46.             drawSHFPBar(origin.translateY(T_HEIGHT_CONST), width,
47.                     SHFP_HEIGHT_CONST)
48.             drawResourcesBar(origin.translateY(T_HEIGHT_CONST + SHFP_HEIGHT_CONST),
49.                     width - RESOURCE_PADDING, + RES_HEIGHT_CONST)
50.     }
51. }
```

Code Listing 4. Pseudocode snippet of how to draw the top portion of an alternative plan.

from the actual stakeholder feature points data (Code Listing 4 line 27). The same method is performed when drawing the boxes representing the initial allocation of resources, but with the width value coming from the list containing actual resource values (as seen in Code Listing 4 line 35).

3.5.1.5. Positioning of Visual Elements

STRATOS fills up the entire area of the screen (preferably in a 16:9 ratio, 1920x1080 pixels and above). The top left corner point, where the coordinates are $x = 0$ and $y = 0$, is the origin point of the whole visualization (labelled as *root_origin* to not be confused with the other origin points given to each visual element). To generate the visualization, the screen has to be vertically divided into the number of alternative plans in the solution set, and horizontally divided into three parts (Code Listing 5).

```
1.  List plans         //list containing all of the plans in the solution set
2.
3.  pWidth = screen_width / the number of plans
4.  height = screen_height / 3
5.  hHeights = List[0, height, height*2]
```

Code Listing 5. Pseudocode snippet of dividing the screen into the spaces to be filled with plans (*pWidth*) and the hierarchy (*hHeights*).

Each of the horizontal sections becomes the designated area for the hierarchy (top to bottom: plans, releases, and then features), while the spaces in between the vertical sections will be filled up by plans. This basic operation subdivides the screen area into equal parts, however, I carefully implemented paddings and other margins to change the aesthetics of the visualization, and improve overall readability. Hence, although I followed these basic operation, the visualizations shown in this thesis may not appear to be subdivided into equal parts. These paddings and margins are left out to simplify the explanation of the algorithm.

The visual elements for plan headers, releases, and features are created after dividing the screen area, and the sizes and positions of each are calculated Code Listing 6). Each of the alternative's plan headers are given an origin point whose x and y coordinates are based on

```
1.  // Note: plans is the list of plans
2.  // releases is the list of releases
3.  // features is the list of features
4.
5.  // calculate positions and sizes for plans and releases
6.  For (i from 0 to the number of plans) {
7.      Plan P = plans[i]
8.      P.origin = Point(root_origin.x + pWidth * i, hHeight[0])
9.      P.width = pWidth - PLAN_PADDING_CONST
10.
11.     int prev_width = 0;
12.     For (j from 0 to the number of releases in P) {
13.             Release R = plans[j].releases
14.             R.origin = Point(P.origin.x + prev_width, hHeights[1])
15.             R.width = toPixels(R.amount_of_resources)
16.             R.height = RELEASE_HEIGHT_CONST
17.     }
18. }
19.
20. // calculate positions and sizes for features
21. For (i from 0 to the number of features) {
22.     Feature F = features[i]
23.     F.origin = Point(root_origin.x + FEATURE_WIDTH_CONST * i, hHeights[2]);
24.     F.width = FEATURE_WIDTH_CONST
25. }
```

Code Listing 6. Calculating the positions and sizes of the plan headers, releases, and features visual elements.

the *pWidth* and *hHeights[0]* (the top of the hierarchy), and a width that is based on the *pWidth.* This width is modified slightly by a padding that prevents the visual elements from appearing too close to each other (see Code Listing 6 lines 8–9). Releases are given an origin point based from the x-coordinate of the origin point of the plan that they belong to and *hHeights[1]*, a width based from the value of the resources flowing into it, and a constant height (Code Listing 6 lines 12–17). Lastly, features are given an origin based on the *root_origin* and *hHeights[2]*, and a constant width (Code Listing 6 lines 21–25). The heights of the bar graphs composing the feature visual element are calculated separately as previously mentioned in section 3.5.1.2. This division of the screen area and placement of visual elements is illustrated in Figure 3.16.

Once the visual elements composing the hierarchy has been laid out, the flow diagrams can be positioned using the positions of the visual elements they connect as anchors. Flows are

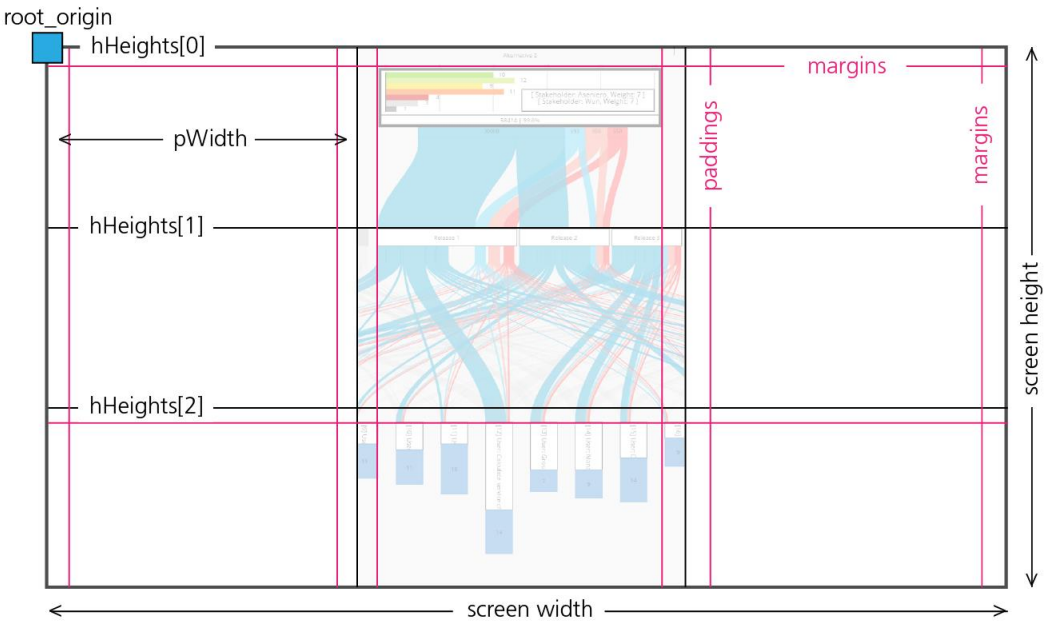


Figure 3.16. Screen division for a solution set with three alternative plans.

created one by one per resource type, connecting each plan header—specifically the part showing the initial allocation of resources—to their respective releases and then to the ones connecting releases to features. For the flow diagrams connecting a plan header to a release, the pixel value of the thickness is calculated from the corresponding resource allocation data of the release. While the thickness of those connecting a release to a feature is calculated from the corresponding required resource data of the feature. **Code Listing 7** on page 59 shows a high level overview of the nested loop operation that performs this task.

Once the positioning of all the visual elements has been done, the visualization can be generated by simply drawing all of the alternative plans, the releases, the flows stored within the releases (flows connecting a plan to the releases), the features, and the flows stored within the features (flows connecting a release to the features).

```
1.  // Note: resources is the list of resources (budget, development time, etc.)
2.  // thickness_of_preceeding_flow is a high level variable for the value of the
3.  // width of the flow immediately to the left of (or before) the current flow
4.
5.  For (i from 0 to the number of plans)
6.      Plan P = plans[i]
7.
8.      For (j from 0 to the number of resources){
9.              Resource S = resources[j]
10.
11.             For (k from 0 to the number of releases in P){
12.                     Release R = P.releases[k]
13.
14.                     // convert the resource allocation data
15.                     thickness = toPixels(R.resource_allocation[S.type])
16.
17.                     // Recall Flow(pointA, pointB, width)
18.                     Point pointA = Point(P.origin.x + thickness_of_preceeding_flow,
19.                                          P.origin.y)
20.                     Point pointB = Point(R.origin.x + thickness_of_preceeding_flow,
21.                                          R.origin.y)
22.                     Flow L = Flow(pointA, pointB, thickness)
23.
24.                     // store the flow in the release
25.                     R.flows.add(L)
26.
27.                     // previous width of a resource flow from release to feature
28.                     int prevWidth_rel = 0
29.                     For (l from 0 to the number of features in R){
30.                             Feature F = R.features[l]
31.                             // convert the required resource data
32.                             thickness = toPixels(F.required_resources[S.type])
33.
34.                             Point pointA = Point(R.origin.x +
35.                             thickness_of_preceeding_flow, R.origin.y + R.height)
36.                             Point pointB = Point(F.origin.x +
37.                             thickness_of_preceeding_flow, F.origin.y)
38.                             F.flows.add(L)
39.
40.                             // store the flow in the feature
41.                             F.flows.add(L)
42.                     }
43.             }
44.     }
45. }
```

Code Listing 7. Pseudocode snippet on how to calculate the positions and thickness of the flows diagrams.

3.6. Chapter Summary

In this chapter, (Section 3.2) I discussed my design process by going over how I applied the design study methodology of Sedlmair et al. (Sedlmair, Meyer and Munzner 2012) in the design and implementation of STRATOS. I outlined the seven design guidelines (*1. Consider all factors of software release planning; 2. Provide a holistic view; 3. Support comparison among alternative plans; 4. Support multiple decision-making strategies; 5. Support details-on-demand; 6. Minimize required interactions; 7. Support individual and collaborative exploration of the data*) that I followed in order to create a visualization to support the decision-making process that takes place in software release planning. In Sections 3.3 and 3.4, I presented the hybrid visualization used in STRATOS which is the end result of the application of the seven design guidelines. I described in detail the different visual elements used for the abstraction of data and the interactions used to allow further examination of the data. Lastly, in Section 3.5, I showed how the hybrid visualization is implemented through a series of pseudocode snippets and illustrations.

In the next chapter, I go over the qualitative method used to evaluate STRATOS.

Chapter 4

STRATOS: STUDY

In this chapter, I describe the qualitative methodology employed to study the scope of STRATOS in supporting the decision-making process involved in software release planning. I highlight the methods used which included the observation of participants' decision-making processes and their behaviours while using STRATOS. I then present the results of the study and discuss its possible implications for future similar visualizations.

This study was done in collaboration with an undergraduate research assistant. It focused on identifying decision-making strategies and examining how they are affected by the use of a visualization of data reflecting the real-life complexity of release planning. This helped in identifying important aspects of the processes used by planners that affect their decision-making. The study also enabled the assessment of whether the design guidelines of STRATOS met the needs of the participants, and whether the visualization helped them arrive to a *good decision*. This was also an opportunity to improve the design guidelines and to find further requirements to help ease the complexity of decision-making in software release planning.

4.1. Methodology

4.1.1. Participants

For the study, participants who have a background in software engineering and release planning were carefully selected through a recruitment process involving the help of the domain expert. The study involved 16 participants; however, one participant data was excluded from the study assessment (Participant 2's knowledge of release planning did not meet the study requirement). As such, 15 participants were assessed (five female and ten male): 12 were students at the graduate level from the University of Calgary—all with computer science, electrical, computer, and or software engineering backgrounds, with some doing research in release planning; and three participants were industry-based software developers whose work involves software release management. They each had different levels of experience with software release planning—nine participants have at least one or more years of experience, and six having less than a year of experience. Two participants had experience in actually using ReleasePlanner™.

4.1.2. Setup

During the study, STRATOS was run on a 72" SMARTboard[3] with a 2K resolution (1988x1080). This was chosen to present the visualization on a large screen that supports touch and pen interaction. This setup, as seen in Figure 4.1, also made it easy to observe each participant's behaviour as they interacted with STRATOS. An HD webcam positioned directly in front of the SMARTboard was used to record each session.

[3] http://education.smarttech.com/en/products/hardware

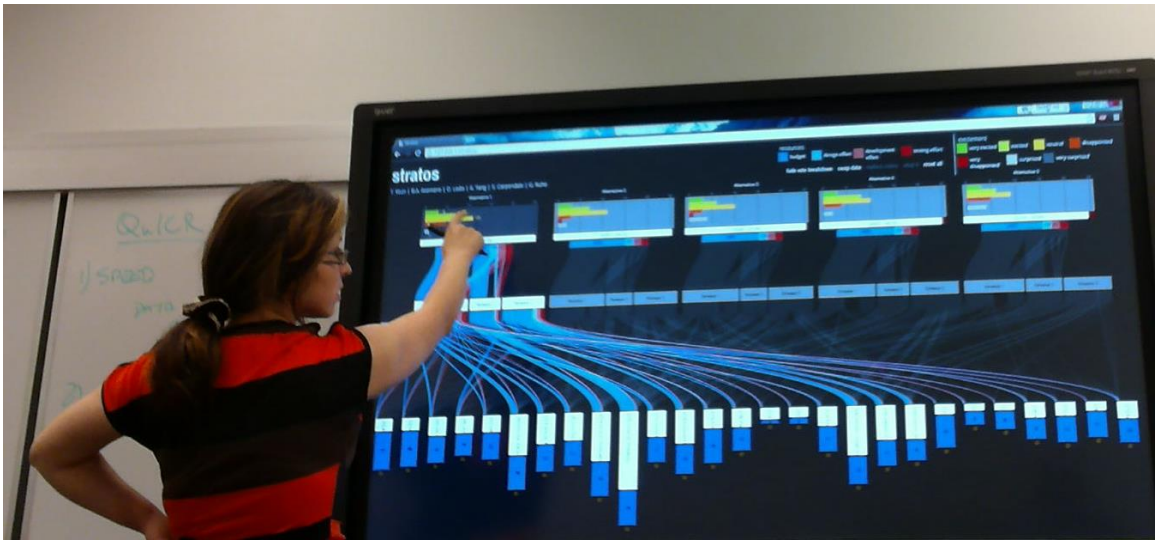


Figure 4.1. A participant interacting with STRATOS during its study.

4.1.3. Procedure

Each participant was run individually through the study, with each session lasting for about an hour. At the beginning of each session, participants were asked to fill out a demographic questionnaire about their expertise and experience with release planning, similar visualizations (e.g. experience with Sankey Diagrams and parallel coordinates), and other visualizations involving software release planning. Afterwards, the participants were given an introduction to STRATOS explaining each component of the visualization and how to interact with it. The remaining part of the study was divided into two phases: a *familiarization phase*, and an *exploration phase*.

4.1.2.1. Familiarization Phase

The purpose of the first phase was to help participants build familiarity with reading and interpreting the hybrid visualization of STRATOS, allowing them to become comfortable interacting with it. In this phase, participants were asked to perform a set of simple tasks with STRATOS. For this, a simple dataset containing a solution set with only three

alternative plans was visualized. The participants were encouraged to ask questions whenever they found something confusing, and assistance or clarification was provided as needed.

The participants were asked to perform the following tasks:

Familiarization Task 1. Naming a feature and stating the amount of resources it requires, as well as any of its dependent features should they exist.

Familiarization Task 2. Reporting to which releases, for each alternative, the feature from Familiarization Task 1 belongs.

Familiarization Task 3. Choosing a release and indicating the amount of features planned to be implemented in its development cycle.

Familiarization Task 4. Finding the feature that requires the greatest amount of resources for its implementation.

Familiarization Task 5. Finding which feature has the top implementation priority according to the stakeholder votes.

Familiarization Task 6. Reporting the amount of resources allocated for an alternative.

Familiarization Task 7. Selecting which alternative has the most positive response from stakeholders.

4.1.2.2. Exploration Phase

In the second phase, the participants were told to take on the role of a project manager (a planner) in an imagined scenario. In this scenario, they were asked to explore a new dataset and choose what they believe to be the optimal plan. For this phase, a more complex solution set with five plans pre-generated from ReleasePlanner™ was used in order to

mimic a real-world scenario. While this dataset was artificial, it reflected real-life situations where each of the alternative plan has trade-offs when compared to one another. A more detailed description of this solution set is provided in the next subsection.

During this phase, the participants interacted with STRATOS on their own. At the same time, they were encouraged to *think aloud* their thoughts as they worked on choosing an optimal plan. Once the participants had chosen an alternative plan, they were asked to summarize their justification as to why they thought their chosen alternative was the best plan in the solution set.

At the end of the study, the participants were asked to fill out a post-study questionnaire about their overall thoughts about STRATOS. Here, they were asked to rate STRATOS on how easy it was to use and read, and their overall confidence on their chosen alternative. They were also asked to list which parts of the visualization helped them the most in making their decision, to give their criticisms, and to make suggestions for improvements.

4.1.4. Exploration Phase Dataset

Table 4.1 shows a simplified overview of the dataset produced by ReleasePlanner™ used in the second phase of our study (the full spreadsheet data is included in Appendix I). The solution set contains five alternatives with several trade-offs. Each alternative contains a different focus;

- **Alternative 1** contains the highest stakeholder satisfaction, as seen from its stakeholder feature points, and contains no *very disappointed* score; however, the number of total features released in this alternative is the lowest.

- **Alternatives 2 and 3** contain the best balance of resources between releases, and are tied for the most number of features released. They both release and postpone the same features, with the only difference given in priority (i.e. some features that are implemented earlier in Alternative 2 are also implemented in Alternative 3 but at a later release).
- **Alternative 4** is similar to Alternative 1; Alternative 4 also contains no disappointment score, albeit having more features released than Alternative 1.
- **Alternative 5** is considered the least optimal plan in the solution set. While it has the second highest amount of features released, it contains the lowest stakeholder

Exploration Dataset Overview

ALTERNATIVES		1	2	3	4	5
Number of implemented features out of 27		16	21	21	18	19
Stakeholder feature points (SHFP)		98.3%	98.2%	97.9%	96.2%	94.9%
Release 1 Insufficient Resources	budget	$0	$0	$0	$0	$0
	time	0 hours	0 hours	0 hours	0 hours	0 hours
Release 1 Extra Resources	budget	$2,200	$600	$500	$2,800	$2,500
	time	109 hours	73 hours	64 hours	119 hours	128 hours
Release 2 Insufficient Resources	budget	$2,200	$600	$500	$2,000	$2,500
	time	0 hours	0 hours	0 hours	0 hours	1 hour
Release 2 Extra Resources	budget	$0	$0	$0	$0	$0
	time	94 hours	114 hours	123 hours	107 hours	87 hours
Resource balance	budget	$0	$0	$0	$800	$0
	time	203 hours	187 hours	187 hours	226 hours	214 hours
"Very Disappointed" Stakeholder points		0	1	1	0	0

Table 4.1. A summarized overview of the potential trade-offs among the alternative plans in the solution set used in the study's exploration phase.

satisfaction based on stakeholder feature points and has a high imbalance in its resource allocation compared to the other alternatives.

Based on EVOLVE II methodology (see Chapter 1 Section 1.1.1), Alternative 2 can be considered to be the hypothetically most balanced plan. This is because Alternative 2 (1) maintains a greater than 95% level of optimality, (2) its stakeholder feature points contains some *surprised* but only one *disappointed* point—overall maintaining a high level of stakeholder excitement, and (3) it has one of the most balanced resource allocation among the alternative plans. Nevertheless, as suggested on Table 4.1, each solution has its own merits; as emphasized in earlier chapters of this thesis, it is precisely because each

alternative have advantages and disadvantages that human involvement in deciding which alternative to choose is imperative.

4.2. Results

The participants did not have any major trouble performing the tasks given to them during the familiarization phase of the study. Although, some had difficulty with the last task (Familiarization Task 7: Selecting which alternative has the most positive response from stakeholders.). This may be because this task asks the participants to find a balance between the different excitement levels (i.e. identifying and selecting between plans that have more *very excited* with some *very disappointed* score, against a plan that has less *very excited* but no *very disappointed* scores) which already involves a form of decision-making. For the remainder of the study, the distribution of participants between their chosen alternative plans was tallied. As seen in Figure 4.2, seven of the 15 participants chose the

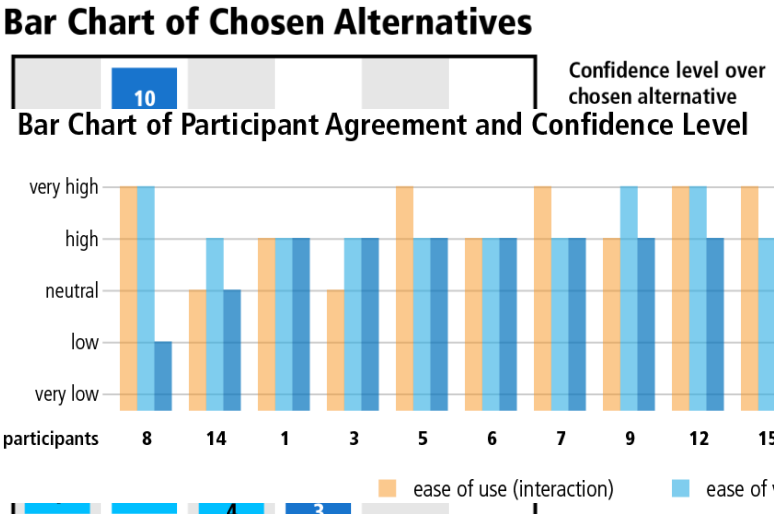


Figure 4.2. A bar chart showing the number of participants (represented by a

Figure 4.3. A bar chart of how each participant agreed according to their perceived ease of use and readability of STRATOS. It is sorted in ascending order based on the participants' confidence level over their chosen alternative.

hypothetically best alternative (Alternative 2), while only one participant (Participant 8) was not able to confidently make a choice. Participant 8's reason for not being able to choose an alternative plan is explained in this chapter's Section 4.5. Assessment of each participant's confidence over the alternative(s) they chose showed that 13 of the 15 positively agreed that they had chosen the best alternative plan as illustrated in Figure 4.3. The same figure also shows most participants agreed that it is easy to interact with and read the hybrid visualization of STRATOS.

Further investigation showed consistency in the participants' justifications of their choices. Alternative 1 was chosen primarily because of its high stakeholder feature points, its zero *very disappointed* stakeholder point, and the total number of features that will be implemented should the plan be put into operation. Alternative 2 was chosen for its balanced resource allocation, its high stakeholder feature points, and, as with Alternative 1, the total number of features that will be implemented with the plan. Alternative 3 was chosen for its balanced resource allocation and the set of features that will be implemented. Alternative 4 was chosen because it implements certain features the participants deemed important but are usually postponed in the other alternatives. Alternative 5 was never chosen; although the participants took note of the amount of features implemented in this plan, they typically dismissed it because of its low stakeholder satisfaction suggesting that the software could still be unsuccessful even if it implements many features.

4.3. Decision Strategies

To confirm that STRATOS supports multiple strategies for decision-making (Design Guideline 4), I sought to find whether the participants employed different strategies while

using STRATOS. Based from the observations of participant behaviours and the common reasons appearing in their justifications, several strategies that they employed in choosing an alternative plan have been identified. While these strategies can each be used on its own, most participants used one strategy as their main justification for choosing an alternative plan, but also used some aspects of the other strategies while interacting with STRATOS.

The decision strategies found are as follows:

Decision Strategy 1. Resource-allocation-based decision strategy

Some participants focused on how the resources are handled within each alternative. They tried to find discrepancies in planned resource allocation such as surplus or insufficient budget.

Decision Strategy 2. Stakeholder-satisfaction-based decision strategy

Some participants focused on how satisfied the stakeholders would be with each alternative. They mostly looked at stakeholder excitement levels. Some also looked at which features were rated highly by the stakeholders and whether those features were scheduled to be released as soon as possible.

Decision Strategy 3. Feature-based decision strategy

Some participants preferred to examine which features are implemented or postponed within each alternative. For example, participant 11 made her own ranking of the features based on the features' priority votes, dependencies, and her personal understanding of the feature based from its description. She then chose the alternative plan that she thought to have implemented more features that were

important to her. Similarly, Participant 8 focused on the feature dependency hierarchy, trying to find which alternative plan prioritizes the features that have dependent features.

Relating these strategies back to the human model of decision-making (Zeleny and Cochrane 1982) I discussed in Chapter 3 Section 3.2, it can be said that the Resource-allocation-based decision strategy is a form of process-oriented approach as it looks at how the resources are allocated and consumed through the software development stages. In contrast, the Stakeholder-satisfaction-based decision strategy and Feature-based decision strategy are more outcome-oriented as they look at what features will be implemented and when, alongside the predicted levels of stakeholder satisfaction.

4.4. Participant Inclination

An important aspect that was identified while reflecting on the observations made during the study is the different *inclinations* that the participants had while interacting with STRATOS. Each participant's behaviour were observed during the study and were categorized accordingly. This was done via open-coding of the observed participant behaviours as seen during the exploration phase of the study and in the recorded videos. In summary, behavioural patterns among the participants were taken note of by focusing on to which elements of the visualization each participant seemed to be most drawn, and any repetitive actions they each made. During a pilot and the study, nine repetitive behaviours were identified (see subsections 4.4.1 and 4.4.2 for these behaviours). A participant was given a mark for a behaviour should they perform it, they are then classified based on which behaviour they had more marks of. The pattern showed that the participants were inclined

to use either *visual* or *numeric cues*, or a mix of both. Of the 15 participants, nine were categorized to have had a visual inclination; three had a numeric inclination; while the other three had a mixed inclination (shown in Figure 4.4). Most participants who had a visual or mixed inclination chose Alternative 2. Those with a numeric inclination were more varied in their choices; however, none of them chose Alternative 2. The choice for Alternative 2 can be attributed from the use of Resource-allocation-based decision strategy because of its balanced resource allocation. Most of the participants also agreed—with Alternative 2's efficient handling of resources being their frequent justification for choosing it. Participants had a visual inclination were also more confident about their choice.

These inclinations are important because they could affect the performance of participants alongside other aspects such as their previous experiences with visualization and expertise in planning. Based on this study, even if two participants used the same decision strategy in choosing an alternative, should they have different inclinations, their method of using STRATOS differed as well. This potentially led them to choose different plans altogether. For example, Participant 3, who had a visual inclination, and Participant 14, who had a numeric inclination, both used a form of Resource-allocation-based decision strategy as their primary strategy but they ultimately chose different plans—with Participant 3 choosing Alternative 4 and Participant 14 choosing Alternative 1. To explain how this could happen, each inclination have been differentiated in the following subsections. The

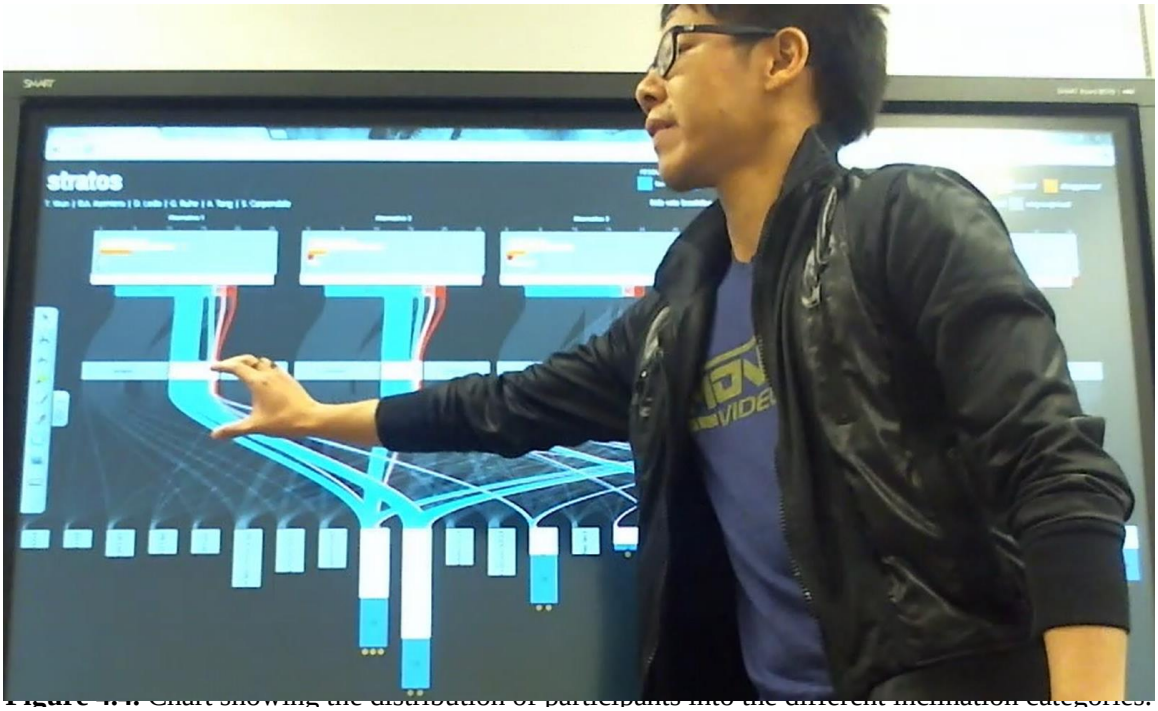


Figure 4.5. A participant with a visual inclination using the flow visual element to approximate resource allocation values.

following subsections also describe the behaviours of the participants who had them, and discuss how STRATOS supported each.

4.4.1. Visual Inclination

Figure 4.5 shows a participant with a visual inclination using STRATOS. Participants with a visual inclination looked at the data primarily using visual cues. They easily understood STRATOS' visualization techniques and used the visual representations to examine and compare alternative plans. The repetitive behavioural pattern of the participants who had this inclination include:

1. Using the perceived widths of the resource allocation flows to compare allocation values among different alternative plans (see Figure 4.5).

2. To compare the releases based on the features they release, they relied on their perceived amount of features released (depicted by highlighted features when a release is selected) as opposed to actually counting the highlighted features, and used the flow visual elements as pointers from a release to the implemented features.
3. Using the heights of the feature visual elements to compare between features.
4. Using the stakeholder satisfaction bar charts to compare stakeholder happiness.
5. Repeated selection of visual elements to highlight and compare factors.

STRATOS' flow visualization was found to effectively support participants who used the Resource-allocation-based decision strategy with this inclination. Those who used this strategy looked at the flow visualization to get an overview of how resources are divided among releases. In particular, they focused on comparing the thickness and gaps between the flow visual elements to find insufficient or surplus resources. Those who used the Feature-based decision strategy used the flow visualization like parallel coordinates; using the flow lines from the features as lines pointing to which releases they are scheduled for implementation for each alternative. Those who used the Stakeholder-satisfaction-based decision strategy mainly used the top portion of the visualization (plan headers) displaying the stakeholder excitement levels, and the stakeholder vote representation at the bottom of each feature.

4.4.2. Numeric Inclination

Participants with a numeric inclination used actual numbers in order to examine and compare alternative plans. These participants used the visual representations only to locate tooltips showing the represented data as numbers on which they heavily relied on. Some even used the SMARTboard pen to write down these numbers to remember them when

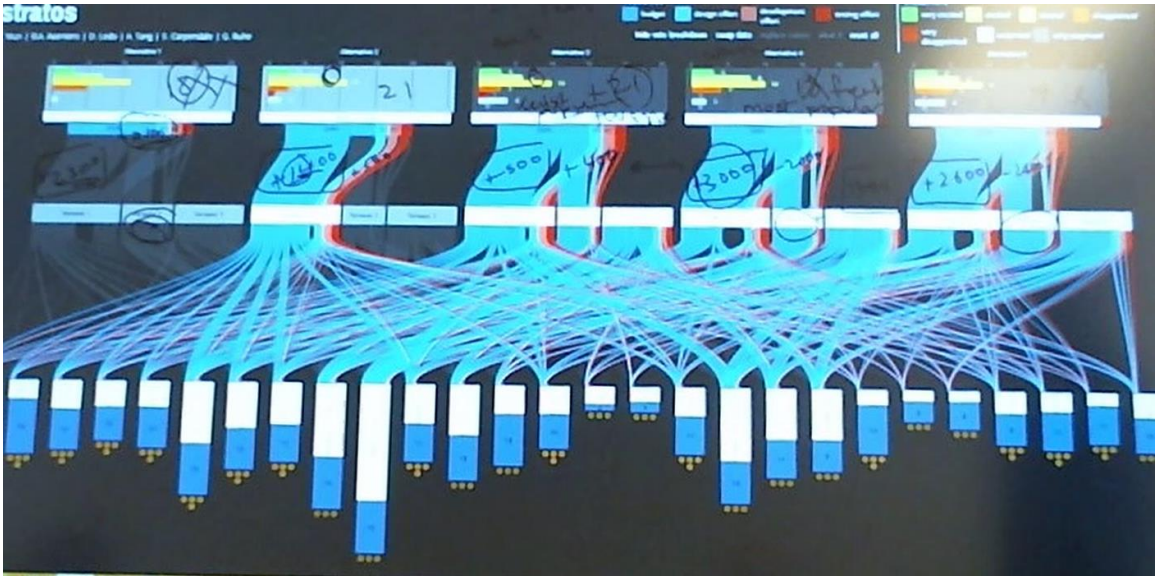


Figure 4.6. Shows a photo of STRATOS after a participant with a numeric inclination used it. The participant took note of the numeric values of the visual element via tooltips and wrote them to help himself remember as he calculated the resource allocation differences among the alternatives.

they needed to calculate. For example, Participant 4 who used the Resource-allocation-based decision strategy wrote down the actual numbers for the resources in each alternative and then calculated the difference between them (see Figure 4.6). The repetitive behavioural pattern of the participants who had this inclination include:

1. Heavy reliance on numeric details given on each visual elements' tooltips.
2. Using the SMARTboard pen to write down numeric details they found from 1.
3. Comparing values through manual computation of differences (i.e. rather than using the perceived widths of the gaps between the resource allocation depicting discrepancies, they instead subtracted the values of the allocated resource and the required resource).
4. Manual counting of features implemented in a release.

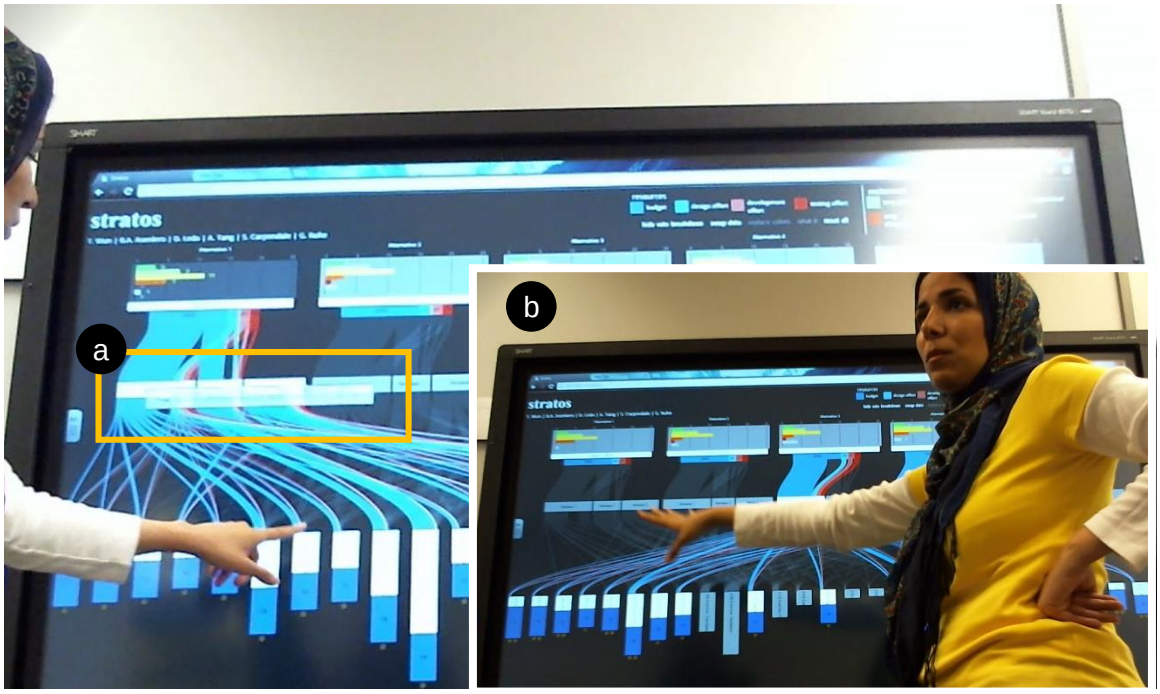


Figure 4.7. A participant with mix inclination using tooltips to see actual numbers (a), while relying on the visual elements to quickly eliminate plans that appear to be less promising (b).

4.4.3. Mixed Inclination

Participants with a mixed inclination used the visual representations similar to those with a visual inclination; however, they also used the actual numeric data to compare visual representations that looked similar. As with visually inclined participants, participants with this inclination were supported effectively by STRATOS. Their inclination also encouraged them to examine the details of the data rather than simply relying on what they have perceived visually (see Figure 4.7). However, they also suffered from the same disadvantages as those with a numeric inclination. The repetitive behavioural patterns of the participants with this inclination are a balanced combination of those listed under visual and numeric inclinations.

4.4.4. Participant Inclinations: General Observations

Most participants who had a visual or mixed inclination chose Alternative 2, while those with a numeric inclination were more varied in their choices; however, none of them chose Alternative 2. This can be attributed to the visualization effectively showing Alternative 2's balanced resource allocation which gave advantage to those who had a visual inclination and used a Resource-allocation-based decision strategy. Evidence for this can be found in the participants' justifications—with Alternative 2's efficient handling of resources being a frequent justification for choosing it. This could also be the reason why participants who were confident with their choice(s) tended to have a visual inclination. Moreover, if a participant thought that the visualization was easy to read, he/she was more likely to be confident in her/his choice(s).

4.5. Discussion and Lessons Learned

The inclinations identified in this study may be artefacts stemming from STRATOS being designed as a visualization tool; however, they could also be a form of personal inclination or preference present in any individual (i.e. something planners will have regardless of the type of decision-making support tool they are using). In this study, participants without previous experience with similar visualizations were more dispersed in their inclinations, while those with previous exposure to similar visualizations were more visually inclined. This suggests that the inclinations stem both from STRATOS' design and from personal experience—meaning that the three inclination categories identified in this study may not be an exhaustive list and further investigation is needed to identify more categories.

Supporting these inclinations should be taken into account in designing future visualization tools like STRATOS.

From the observations, while it may be that most of the participants came in to the study with the resource-allocation-based decision strategy as their pre-established strategy for decision-making, it was interesting to see a connection with the use of this strategy and the way STRATOS is designed. Because STRATOS' main visualization is a flow diagram of resources, it can be argued that participants who have a visual or mixed inclination have found it easier to assess each alternative based on their resource allocation. However, it is also possible that they may have been urged to employ this strategy as their main decision strategy because the most prominent visual cue in STRATOS' visualization is the flow diagram of resources. As release planning is often subjective due to the multitude of constraints and stakeholder values, it is good that STRATOS' visualization allowed for the exploration of the different alternatives. What matters is that none of the participants chose Alternative 5, which was deemed to be the least balanced plan in the solution set. Thus, despite the fact that some participants did not choose Alternative 2, they were able to make informed decisions, and the visualization did not lead them towards a detrimental solution. This shows that the visualization showcased the different alternatives and did not bias the participants to fixate on one alternative.

4.5.1. Discussion Regarding the Design Guidelines

Relating these findings back to STRATOS' design guidelines (Chapter 3, Section 3.2.1), I found that all strategies employed by the participants involved some form of examining all of the factors of release planning—suggesting that STRATOS' visual elements allowed the planners to examine and consider as many release planning factors as they could (Design

Guideline 1). However, while the basic factors of release planning were easily comparable in our approach, a factor that is not immediately apparent is the risk factor. An automated identification and presentation of risk factors in each alternative will greatly help planners in risk assessment to find plans that are less risky for the development team.

The unified layout of STRATOS showed all of the factors and how they relate with each other, providing a holistic view (Design Guideline 2) and allowing participants to compare alternatives (Design Guideline 3). However, the study showed that while a singular layout could help in alleviating the mental load coming from view switching, this gives its own type of mental load and that a considerable amount of training time is necessary for the participants to be able to use STRATOS comfortably and effectively. This issue raises a question on whether compartmentalizing the single view into several visualization widgets (one visualization per factor) and updating all whenever one is interacted with by the planner, could lower the mental load. This requires further investigation as multiple view switches have also been shown to also cause cognitive overload (Wang Baldonado, Woodruff and Kuchinsky 2000). Arriving at a good balance would require additional future work. Nevertheless, the choice of combining Sankey diagrams and parallel coordinates in a tree view hybrid visualization proved to be useful in supporting participants who had a visual or mixed inclination.

To some extent, STRATOS also supported those with a numeric inclination via details-on-demand (Design Guideline 5). STRATOS provides the ability to drill down to actual data through tooltips—and numerically inclined participants did use the structure of STRATOS to find the right tooltips—there are some aspects of this numerical approach that the design of STRATOS did not effectively support. For example, participants who used a Stakeholder-

satisfaction-based decision strategy with a numeric inclination were not able to draw out numbers that compose the stakeholder excitement levels—data that they needed to see. On the other hand, Participant 8, who used a Feature-based decision strategy—focusing mostly on stakeholder votes for each features, dependent features, and when they are implemented—had to go through each of the features to see their details in the tooltips. This put a heavy mental load on the participant which eventually led him to say that he could not use STRATOS to find the most optimal plan.

It is hard to say whether these difficulties are due to the participants not understanding the visualization enough, or because some type of support was lacking, as our post-study questionnaire data showed that most numerically inclined participants still agreed that the visualization was easy to read (see Figure 4.3 on page 68). Nevertheless, rather than having the participants dig for numerical information through tooltips, it is advisable to specifically design ways of how to integrate numerical data within the same view of the visualization. Minimizing the required interactions for all the inclinations is imperative to ensure they are all well-supported (Design Guideline 6). This minimization of required interactions can be extended to the feature visual elements as well. For instance, rather than showing a single bar for the stakeholder votes with its breakdown shown in the tooltip, the domain expert suggested that it would be more meaningful to see the vote breakdown as separate parts of the stacked bar. This is because some stakeholders have a higher weight than others, and seeing the vote of a targeted stakeholder at-a-glance can add more utility to the visualization. The steps required in finding the dependencies of features can also be minimized by arranging the features in space based on their dependencies, extending the tree layout of the hybrid visualization, rather than keeping them in a single axis.

In Section 4.3, the hybrid visualization of STRATOS has been shown to support multiple decision-making strategies (Design Guideline 4). However, this study focused on individual analysis, hence, further evaluation is needed to identify group decision-making strategies and how support for individual decision-making strategies can be appropriated to support group dynamics (Design Guideline 7).

4.6. Chapter Summary

In this chapter, I have described the qualitative methodology employed to evaluate the effectiveness of STRATOS in supporting the decision-making process involved in software release planning. In Section 4.1, I gave a detailed description of the setup and procedure of the study, and an overview of the visualized dataset. I have presented the results of this study in Section 4.2 showing that STRATOS did help the participants arrive at what was considered a good decision.

I then outlined the decision strategies (Resource-allocation-based decision strategy, Stakeholder-satisfaction-based decision strategy, and Feature-based decision strategy) employed by the participants in choosing the most optimal plan and how each was supported by STRATOS. I also outlined the participant inclinations (Visual Inclination, Numeric Inclination, and Mixed Inclination) that had an effect on their method of using STRATOS and the previously mentioned strategies.

Lastly in Section 4.5, I discussed the outcome of the study, emphasizing the strengths and weaknesses of STRATOS—with regards to the validity of its design guidelines—and foreshadowing possible implications for future visualizations that aim to give decision-making support in software release planning.

In the next chapter, I go over some of the possible future work stemming from the lessons learned in this study and give a conclusion to this thesis.

Chapter 5

CONCLUSION

This final chapter serves as a closing remark to this thesis in which the ideas presented in the previous chapters are recapitulated and concluded. Here, I also explore some future work which includes improvements and ideas beyond the scope of this thesis.

5.1. The Work Thus Far

In Chapter 1, I gave a background information about the practice of software release planning, and shed light on why a well-informed decision is vital for planners to be able to choose the optimal plan for releasing software into market. In this chapter, I establish that this thesis is primarily about supporting planners' decision-making processes through visualizations that enable them to make informed decisions. Thus, this thesis is concerned specifically with ***how to support planners in choosing an optimal plan by visualizing the interrelated factors of release planning.***

5.1.1. Revisiting the Thesis Questions

In order to support planners in choosing an optimal plan by visualizing the interrelated factors of release planning, I focused on exploring solutions to three research questions, each dictated by an underlying problem or challenge planners face in making decisions. These research questions are reiterated below, along with the description of the underlying problem which dictated its exploration:

Research Question 1 is motivated by *Problem 3: It can be difficult for planners to compare alternative plans in order to be able to choose the best one.* This problem suggests that for a visualization to effectively support decision-making in software release planning, it must simplify the planners' task of comparing alternative plans in a solution set. Hence, Research Question 1 is ***how can a visualization be designed such that it helps planners see the trade-offs between plans at-a-glance?***

Research Question 2 is motivated by *Problem 2: It can be difficult for planners to account for the interrelated factors of software release planning.* This problem suggests that for a visualization to effectively support decision-making in software release planning, it must help the planners account for as many as possible interrelated factors of software release planning. Hence, Research Question 2 is ***how can a visualization be designed such that it visualizes the interrelatedness of the different factors of release planning?***

Lastly, Research Question 3 is motivated by *Problem 1: Planners can have different decision-making processes.* This problem suggests that to effectively support decision-making in software release planning, visualizations must be able to support different types of decision-making processes. These processes can be in a form of outcome-oriented or

process-oriented approaches (Chapter 3, Section 3.2). Hence, Research Question 3 is ***how can a visualization be designed such that it supports multiple types of decision-making processes among different planners?***

5.1.2. Proposed Solution

Chapter 3 outlined the exploratory solution this thesis offers to solve these questions. Also in this chapter, I described how the design study methodology was applied to develop a set of visualization design guidelines exemplified by a visualization—STRATOS (STRATegic software release planning Oversight Support). These design guidelines were:

Design Guideline 1. Consider as many as possible factors.

Knowing that the conditions of multiple factors of software release planning is important for planners to be able to make good and well-informed decisions, the design of the visualization must take into account visualizing as many factors as possible.

Design Guideline 2. Provide a holistic view.

Visualizations for supporting decision-making in software release planning should not only be able to show the factors but must also be able to show how they relate to one another. A holistic view allows decision makers to consider most of the factors with considerable ease rather than trying to do so while switching between multiple views.

Design Guideline 3. Support comparison among alternative plans.

Comparing trade-offs among possible alternatives is at the heart of decision-making in software release planning. Therefore, plans must be shown as distinct visual elements within the visualization to help planners easily identify them as

alternatives to one another. At the same time, consistency across representations should be employed such that they could be visually compared.

Design Guideline 4. Support multiple decision-making strategies.

Different planners often have different approaches on deciding what the best alternative is in regards to their project's goal. An interactive visualization should allow planners to explore the data according to their own preferences; by letting them find possible outcomes (outcome-oriented approach) and or by helping them better understand a given solution (process-oriented approach).

Design Guideline 5. Support details-on-demand (Shneiderman, The Eyes Have it: A Task by Data Type Taxonomy for Information Visualizations 1996)**.**

While visually conveying information allows planners to do simple comparisons at-a-glance, they should still be able to access detailed information such as the numeric values of the visualized data. This could help planners to accurately distil information that look similar when visualized.

Design Guideline 6. Minimize required interactions.

Minimizing interaction over-head by avoiding excessive clicking, selecting, etc., while still providing full visualization and data access will make interacting with the visualization more pleasant. This could lead to better acceptance of the tool, making it easier to be integrated with other support tools or methods that the planners may already using.

Design Guideline 7. Support individual and collaborative exploration of the data.

Release planners may explore alternatives individually or in a group, such as when having a meeting. Hence, there is an advantage to allow planners—either

individually or as a group—to explore the visualization simultaneously according to their own practices and as a communicative tool.

In summary, Design Guideline 3 and 6 are means of providing a solution to Research Question 1, Design Guideline 2, as well as 1 and 5 are means of providing a solution to Research Question 2, and lastly, Design Guideline 4, as well as 1 and 5–7 are means of providing a solution to Research Question 3. A more detailed description of how these guidelines explores the research questions of this thesis was provided on Chapter 3 Section 3.2.1. These design guidelines were realized in STRATOS, a hybrid visualization formed by combining aspects of Sankey diagrams and parallel coordinates within a multiple tree layout. A full description of this visualization tool was given in Chapter 3 Sections 3.3–4, while pseudocode descriptions of its drawing algorithm was given in Section 3.5.

To assess the scope a visualization like STRATOS can potentially support, a qualitative study was conducted as described in Chapter 4. Several decision strategies supported by STRATOS were identified through this study showing that a visualization created under the design guidelines offered in this thesis could support multiple decision-making strategies. Moreover, the study results also suggested that STRATOS enabled the planners to explore the trade-offs between the alternative plans, helping them arrive to a good decision. The decision strategies identified through the study are reiterated as follows:

Decision Strategy 1. Resource-allocation-based decision strategy

A strategy employed by participants who focused on how the resources are handled within each alternative, finding discrepancies in the use of resources.

Decision Strategy 2. Stakeholder-satisfaction-based decision strategy

A strategy employed by participants who focused on how happy the stakeholders would be with each alternative.

Decision Strategy 3. Feature-based-decision strategy

A strategy employed by participants who preferred to examine which features are implemented or postponed within each alternative.

The qualitative study also allowed for the identification of different participant inclinations that affected their use of STRATOS. From the findings, it can be said that these inclinations stem from the way the visualization has been designed and from personal inclination or preference, and so, should be taken into consideration when developing future visualizations like STRATOS. A more thorough description of these inclinations and a discussion on how they affected the participants' decision-making processes was detailed in Chapter 4 Sections 4.4 and 4.5 respectively.

The participant inclinations are reiterated as follows:

Participant Inclination 1. Visual Inclination: A tendency for participants to examine and compare alternative plans in the solution set primarily using visual cues.

Participant Inclination 2. Numeric Inclination: A tendency for participants to examine and compare alternative plans in the solution set through actual numbers and manual computations.

Participant Inclination 3. Mixed Inclination: A tendency for participants to use a balance between the previously described inclinations.

5.1.3. Contributions

As previously stated in Chapter 1, the contributions of this thesis are as follows:

Thesis Contribution 1. STRATOS, a hybrid visualization that visualizes potential plan outcomes and reveals the decision-making factors for several plans within a single view, making it possible to compare several plans at once.

Thesis Contribution 2. The qualitative evaluation methodology employed to study how planners used STRATOS and possibly similar visualizations

Thesis Contribution 3. The results of the study of STRATOS and its possible implications for other visualizations supporting decision-making in software release planning.

5.2. The Work Ahead

Avenues for future work can be derived from the lessons learned from the study of STRATOS as discussed in Chapter 4 Section 4.5. In the same section, the limitations of STRATOS, its design guidelines, and the qualitative method used in its study were addressed, foreshadowing some of the improvements that can be made. These improvements include scaling STRATOS to overcome the limitations of its current state, and performing further evaluation and investigation to answer some of the questions that were left unanswered.

5.2.1. Improving STRATOS

STRATOS can be improved in many ways. One way is to improve support for planners with a numeric inclination. As stated in Chapter 3 Section 3.2, I worked closely with a domain expert over the course of developing STRATOS. The domain expert suggested that in order

to better support this type of inclination, a type of dashboard that integrates the visual elements of STRATOS with a view of the spread sheet containing the numerical data. Dashboards have been shown to provide better awareness of both high-level and low-level aspects of data (Treude and Storey 2010), making its incorporation with the visualization of STRATOS convincingly beneficial.

As I have pointed out in Chapter 4 Section 4.5, while STRATOS' singular layout alleviated the mental load stemming from view switching, it also came with its own type of mental load. That is, some participants felt overwhelmed by the amount of information being shown at once, requiring participants to undergo a considerable amount of training time. As stated Section 4.5.1, a possible solution is to compartmentalize the different parts of the visualization into several *visualization widgets* (possibly one factor per widget). The incorporation of dashboard elements with STRATOS' visualization could help in compartmentalizing the different parts of its singular view. The domain expert also

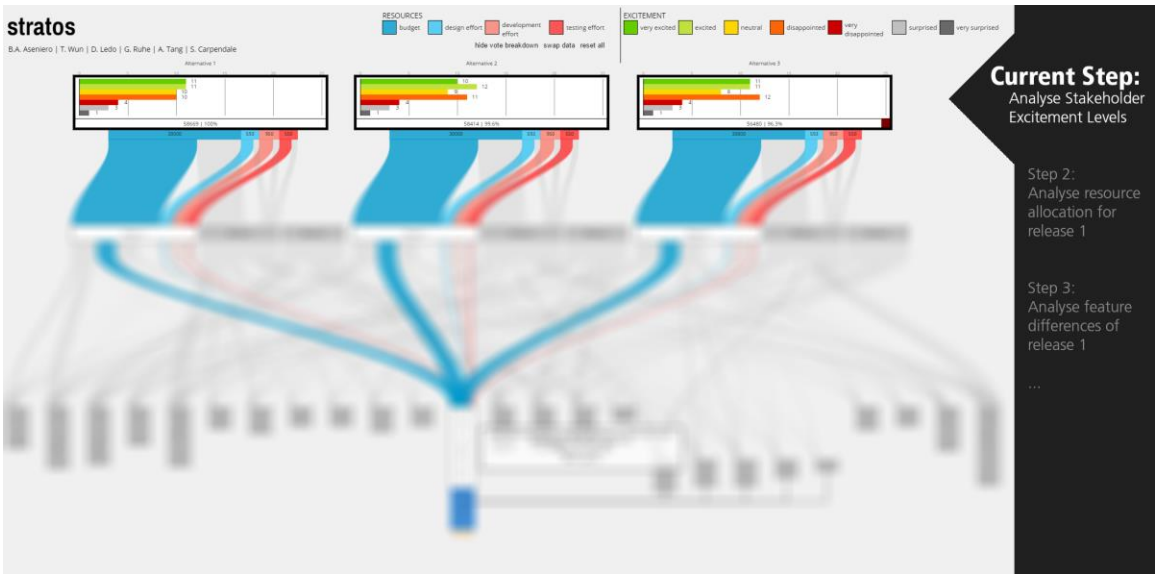


Figure 5.1. Implementing a step-by-step guide the planners could follow while using STRATOS. The parts of the visualization that are currently not needed for the step are blurred out.

suggested implementing a feature that guides the planner along a step-by-step analysis of the data shown in the visualization (see Figure 5.1). This would enable the planner to focus on a certain portion of the visualization, while the rest of the visualization is rendered out-of-focus to reduce visual clutter but maintain overall awareness. For example, if a planner is currently at the step that tells him/her to compare the stakeholder happiness among the alternative plans, then the visualization will focus on each alternatives' header and blur out the rest of the visualization. This feature should also allow for being overridden by the planner to retain the perceived *freedom of choice* afforded by STRATOS' design.

There are also other features that could be implemented to turn STRATOS from a simple visualization tool into a full visual analytics tool. *Direct manipulation*, or being able to modify the data on the spot (Shneiderman and Maes, Direct Manipulation vs. Interface Agents 1997) could further increase decision support. For example, a planner can change the values for a resource such as budget by dragging the width of the bar representing the resource; and after doing so, the visual elements update accordingly, adjusting to the new input value. Furthermore, embedded analysis of the data could highlight areas of the data that could be missed due to human error.

STRATOS' drawing algorithm can also be improved through optimization. While the current implementation is sufficient for running locally on a single computer, optimizing its code will be beneficial when it is extended to support non-collocated interaction with more than one people over a network or the internet. To support this, one may look into the suggestions given by Gutwin and Greenberg from the groupware toolkit (Gutwin and Greenberg 1995). This includes providing awareness of what the other members of the team are doing through a multi-cursor visualization of each member's cursor (i.e. cursor

movements and interactions performed by a member is reflected on all members' view), and providing video conferencing.

5.2.2. Further Evaluation and Investigation

One limitation of the qualitative study employed to study STRATOS is that the majority of its study participants were students, and only a handful were planners from industry. While a participant sample is seldom perfect and this is always a limitation, this study's participants' skill set was both representative and sufficient for the study's purpose. The goal of this study was to understand the *scope* of what a visualization like STRATOS potentially supports, and not whether STRATOS (in its current prototype form) should be the tool used in industry. Nevertheless, a study involving industry planners, such as project and product managers, could shed light into how visualizations like STRATOS could perform in the wild. It could also lead to the identification of other decision-strategies, inclinations, and inform some leeway on how this type of visualizations can be integrated with existing management practices such as Kanban. Furthermore, the study was performed individually between participants, and as such, it did not investigate what roles STRATOS could take on as part of a development team dynamics. Investigating this could inform us about group decision-making strategies, how they relate to individual decision-strategies and inclinations, and how to best support them.

5.3. Closure

In conclusion, the decision-making process that takes place in software release planning can be supported through visualization. In particular, this thesis has done so through ***supporting planners in choosing an optimal plan by visualizing the interrelated factors***

of release planning. Through the design guidelines exemplified by STRATOS, I have shown the process of designing a visualization that supports this decision-making process, arriving at the end result of a hybrid visualization combining Sankey diagrams and parallel coordinates in a multiple tree layout. To support this claim, a qualitative study has been performed to study STRATOS, arguably showing that by following our design guidelines, the resulting visualization was able to support multiple types of decision-making processes and inclinations. This thesis can perhaps encourage the development of other visualization tools that provide decision-making support—and in the future, extend the lessons learned from here to the design of visualizations supporting decision-making beyond software release planning.

BIBLIOGRAPHY

Amandeep, Guenther Ruhe, and Mark Standford. 2004. "Intelligent Support for Software Release Planning." In *Product Focused Software Process Improvement*, 248-262. Springer Berlin Heidelberg.

Anderson, David J. 2010. *Kanban.* Blue Hole Press.

Bhawnani, Pankaj, and Guenther Ruhe. 2005. "ReleasePlanner–Planning New Releases for Software Maintenance and Evolution." *ICSM (Industrial and Tool Volume).* 73–76.

Buja, Andreas, John Alan McDonald, John Michalak, and Werner Stuetzle. 1991. "Interactive Data Visualization using Focusing and Linking." *IEEE Conference on Visualization.* San Diego, CA: IEEE. 156–163.

Card, Stuart K., Jock D. Mackinlay, and Ben Shneiderman. 1999. *Readings in Information Visualization: Using Vision to Think.* Morgan Kaufmann.

Carlshamre, Pär, Kristian Sandahl, Mikael Lindvall, Björn Regnell, and Johan Natt och Dag. 2001. "An Industrial Survey of Requirements Interdependencies in Software Product Release Planning." *Fifth IEEE International Symposium on Requirements Engineering.* IEEE. 84–91.

Clark, Wallace, and Henry Laurence Gantt. 1923. *The Gantt Chart, a Working Tool of Management.* New York: Ronald Press.

Feather, Martin S., Steven L. Cornford, James D. Kiper, and Tim Menzies. 2006. "Experiences using Visualization Techniques to Present Requirements, Risks to Them, and Options for Risk Mitigation." *Requirements Engineering Visualization REV.* IEEE.

Feiner, Steven. 1988. "Seeing the Forest for the Trees: Hierarchical Displays of Hypertext Structures." *ACM Conference on Office Information Systems.* New York, NY, USA: ACM. 205–212.

Fekete, Jean-Daniel, Jarke J. van Wijk, John T. Stasko, and Chris North. 2008. "The Value of Information Visualization." In *Information Visualization*, edited by Kerren Andreas, John T. Stasko, Jean-Daniel Fekete and Chris North, 1–18. Springer Berlin Heidelberg.

France, Robert, Andy Evans, Kevin Lano, and Bernhard Rumpe. 1998. "The UML as a Formal Modeling Notation." *Computer Standards & Interfaces* 325–334.

Greer, D., and Guenther Ruhe. 2004. "Software Release Planning: an Evolutionary and Iterative Approach." *Information and Software Technology* 243–253.

Gutwin, Carl, and Saul Greenberg. 1995. "Support for Group Awareness in Real Time Desktop Conferences." *Proceedings of The Second New Zealand Computer Science Research Students Conference* . Hamilton, New Zealand. 18–21.

Henry, Nathalie, Jean Daniel Fekete, and Michael J. McGuffin. 2007. "NodeTrix: A Hybrid Visualization of Social Networks." *IEEE Transactions on Visualization and Computer Graphics* 1302–1309.

Inselberg, Alfred, and Bernard Dimsdale. 1990. "Parallel Coordinates: A Tool forVisualizing Multi-dimensional Geometry ." *First Conference on Visualization.* Los Alamitos, CA, USA: IEEE Computer Society Press. 361–378.

Jantunen, Sami, Laura Lehtola, Donald C. Gause, U. Rex Dumdum, and Raymond Barnes. 2011. "The Challenge of Release Planning." *Proceedings of the Fifth International Workshop on Software Product Management (IWSPM).* Trento, Italy: IEEE. 36–45.

Kosara, Robert, Fabian Bendix, and Helwig Hauser. 2006. "Parallel Sets: Interactive Exploration and Visual Analysis of Categorical Data." *IEEE Transactions on Visualization and Computer Graphics* 558–568.

Lurie, Nicholas, and Charlotte Mason. 2007. "Visual Representation: Implications for Decision Making." *Journal of Marketing* 160–177.

Piegl, Les, and Wayne Tiller. 1995. *Curve and Surface Basics.* Springer Berlin Heidelberg.

Reihmann, Patrick, Manfred Hanfler, and Bernd Froehlich. 2005. "Interactive Sankey Diagrams." *IEEE Symposium on Information Visualization INFOVIS.* IEEE. 233–240.

Ruhe, Guenther. 2011. *Product Release Plannning: Methods, Tools, and Applications.* CRC Press.

Rumbaugh, James, Ivar Jacobson, and Grady Booch. 2004. *The Unified Modeling Language Reference Manual.* 2nd. Pearson Higher Education.

Sankey, H. R. 1896. "The Thermal Efficiency of Steam-Engines." *Minutes of the Proceedings* 182–212.

Schmidt, Mario. 2008. "The Sankey Diagram in Energy and Material Flow Management." *Journal of Industrial Ecology* 82–94.

Sedlmair, Michael, Miriah Meyer, and Tamara Munzner. 2012. "Design Study Methodology: Reflections from the Trenches and the Stacks." *IEEE Transactions on Visualization and Computer Graphics* 2431–2440.

Shneiderman, Ben. 1996. "The Eyes Have it: A Task by Data Type Taxonomy for Information Visualizations." *IEEE Symposium on Visual Languages.* IEEE. 336–343.

Shneiderman, Ben, and Pattie Maes. 1997. "Direct Manipulation vs. Interface Agents." *Interactions* 42–61.

Treude, Cristoph, and Margaret-Anne Storey. 2010. "Awareness 2.0: Staying Aware of Projects, Developers and Tasks using Dashboard and Feeds." *ACM/IEEE International Conference on Software Engineering.* Cape Town, South Africa: IEEE. 365–374.

Tufte, Edward R. 1983. *The Visual Display of Quantitative Information.* Cheshire, CT, USA: Graphics Press.

Wang Baldonado, Michelle Q., Allison Woodruff, and Allan Kuchinsky. 2000. "Guidelines for Using Multiple Views in Information Visualization." *Proceedings of the Working Conference on Advanced Visual Interfaces.* New York, NY, USA: ACM. 110–119.

Wnuk, Krzysztof, Björn Regnell, and Lena Karlsson. 2008. "Visualization of Feature Survival in Platform-Based Embedded Systems Development for Improved Understanding of Scope Dynamics." *Requirements Engineering Visualization REV.* IEEE. 41–50.

Zeleny, Milan, and James L. Cochrane. 1982. *Multiple Critera Decision Making.* New York: McGraw-Hill.

Appendix I

EXPLORATION PHASE DATASET

Appendix 1 contains the full spreadsheet data visualized during the exploration phase of STRATOS' study.

Solution Set		Alternatives				
ID	Project	1	2	3	4	5
1	Account Creation	1	1	1	1	1
2	Login	1	1	1	1	1
3	List of Lessons + Navigation	1	1	1	1	1
4	Lesson Progress	3	1	1	3	1
5	List of Exercises	1	1	1	1	1
6	Exercise: Fill in the Blanks	1	1	1	1	1
7	Exercise: Multiple Choice	1	1	2	1	1
8	Exercise: True or False	1	1	2	1	1
9	Exercise: Translation	2	2	2	2	2
10	Exercise: Dictation	2	3	3	2	2
11	Grammar Summary	1	1	1	3	3
12	Exercise Feedback	3	3	3	3	3
13	Exercise: Correction	2	3	3	2	2
14	Import Lesson	3	1	1	3	3
15	Export Progress	3	3	3	3	2
16	Import Progress	3	2	2	3	1
17	Admin: Create Lesson	1	1	1	1	1
18	Admin: Create Exercise	3	2	1	3	3
19	Admin: Create Question for Exercise	3	3	3	3	3
20	Admin: Attach Media	3	3	3	3	3
21	Admin: See student progress	1	1	1	1	1
22	Admin: Add student	2	2	1	1	1
23	Admin: Remove Student	1	2	1	1	3
24	Admin: Delete Question	3	1	2	1	1
25	Admin: Delete Exercise	1	1	1	1	1
26	Admin: Delete Lesson	1	1	1	1	1
27	Admin: Save lesson set	1	2	1	1	3

Spreadsheet 1. Shows the spreadsheet of the solution set containing five alternative plans and the distribution of the 27 features into the releases among each alternative.

Alternative	degree of optimality	SHFP
1	98.3	35430
2	98.2	35387
3	97.9	35285
4	96.2	34692
5	94.9	34183

Spreadsheet 2. Shows the spreadsheet data containing the stakeholder feature points (SHFP) and degree of optimality for each alternative plan.

Requirement					Precedence & Coupling Constraints		Resource Consumption			
ID	Group	Requirement	Description	Pre-Assignment	Precedes	Coupled to	Budget	Design Effort	Development Effort	Testing Effort
1	Login	Account Creation	students or instructors are able to create an account		03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,		600	6	10	3
2	Login	Login	Students and instructors can log into their accounts		03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,		600	4	5	3

ID	Group	Requirement	Description	Pre-Assignment	Precedes	Coupled to	Budget	Design Effort	Development Effort	Testing Effort
3	Exercises	List of Lessons + Navigation	List of lessons to be studied is presented, each lesson with a set of exercises.		04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,		1000	13	16	15
4	Exercises	Lesson Progress	Lesson progress (lessons accomplished / total number of lessons) is presented.				900	12	8	3
5	Exercises	List of Exercises	Each lesson features a set of exercises to be solved.		07, 08, 09, 10, 11, 12, 13,		900	5	10	3

ID	Group	Requirement	Description	Pre-Assignment	Precedes	Coupled to	Budget	Design Effort	Development Effort	Testing Effort
6	Exercises	Exercise: Fill in the Blanks	A sentence is provided, in which the user fills in the information required to properly complete a sentence. This may include modifying words in brackets to the proper tense.		12, 13,		2500	25	30	20
7	Exercises	Exercise: Multiple Choice			12, 13,		1800	22	20	10
8	Exercises	Exercise: True or False			12, 13,		1600	18	20	10
9	Exercises	Exercise: Translation	Users are required to translate a one or more sentences.		12, 13,		3000	25	60	40
10	Exercises	Exercise: Dictation	Users listen to an audio and are required to type what they hear.		12, 13,		5000	30	60	20

ID	Group	Requirement	Description	Pre-Assignment	Precedes	Coupled to	Budget	Design Effort	Development Effort	Testing Effort
11	Exercises	Grammar Summary	Users are presented with a summary of the grammar points required for the current exercise set.				1500	20	22	5
12	Exercises	Exercise Feedback	Users are able to go through their answers and review whether they answered correctly or incorrectly. A score is provided.				2000	30	17	15
13	Exercises	Exercise: Correction	Users are able to correct their exercises. Grade is adjusted accordingly.				1000	18	17	7

ID	Group	Requirement	Description	Pre-Assignment	Precedes	Coupled to	Budget	Design Effort	Development Effort	Testing Effort
14		Import Lesson	Users can import a full chapter containing grammar summaries and exercises and add them to a current set.				1000	14	15	4
15		Export Progress	Users can save all the information associated to their account and their progress.				500	8	8	4
16		Import Progress	Users can open all their account information. This is useful when migrating between machines.				400	4	4	4
17	Admin	Admin: Create Lesson	Instructors can create a new lesson, which holds exercises.		04, 05,		1000	20	22	9
18	Admin	Admin: Create Exercise	Instructors can create their own exercise.		19,		3000	30	32	14

ID	Group	Requirement	Description	Pre-Assignment	Precedes	Coupled to	Budget	Design Effort	Development Effort	Testing Effort
19	Admin	Admin: Create Question for Exercise	Allows administrators to add a question to the exercise.		20,		2000	20	20	10
20	Admin	Admin: Attach Media	Allows instructors to attach media.				2000	10	28	7
21	Admin	Admin: See student progress					400	8	8	2
22	Admin	Admin: Add student	Instructors can create an account for students instead of them having to create it themselves.				200	5	5	3
23	Admin	Admin: Remove Student					200	2	2	1
24	Admin	Admin: Delete Question	Deletes a single question from the exercise.				700	10	7	7

ID	Group	Requirement	Description	Pre-Assignment	Precedes	Coupled to	Budget	Design Effort	Development Effort	Testing Effort
25	Admin	Admin: Delete Exercise					600	10	10	4
26	Admin	Admin: Delete Lesson					300	7	6	2
27	Admin	Admin: Save lesson set	Saves the lesson set, which can be later imported by a student or instructor.				800	13	18	8

Spreadsheet 3. Shows the spreadsheet data containing the information about the features of the software. This includes the feature ID, name, description, dependent features, precedent features, and resource requirements.

Value Voting		Stakeholders	
ID	Requirement	Aseniero	Ledo
1	Account Creation	9	8
2	Login	9	7
3	List of Lessons + Navigation	5	7
4	Lesson Progress	3	7
5	List of Exercises	5	7
6	Exercise: Fill in the Blanks	7	8
7	Exercise: Multiple Choice	7	5
8	Exercise: True or False	7	4
9	Exercise: Translation	7	8
10	Exercise: Dictation	7	8
11	Grammar Summary	5	6
12	Exercise Feedback	7	6
13	Exercise: Correction	9	6
14	Import Lesson	5	5
15	Export Progress	1	1
16	Import Progress	1	2
17	Admin: Create Lesson	7	4
18	Admin: Create Exercise	7	5
19	Admin: Create Question for Exercise	7	4
20	Admin: Attach Media	5	4
21	Admin: See student progress	9	5
22	Admin: Add student	3	3
23	Admin: Remove Student	3	5
24	Admin: Delete Question	7	2
25	Admin: Delete Exercise	7	4
26	Admin: Delete Lesson	7	4
27	Admin: Save lesson set	7	3

Spreadsheet 4. Shows the spreadsheet containing the values of stakeholder priority votes on each feature.

Resources			Resource Capacities	
Resource Name	Resource Type	Resource Units	Release 1	Release 2
Budget	Budget	dollars	16000	7000
Design Effort	Effort	hours	230	153
Development Effort	Effort	hours	246	161
Testing Effort	Effort	hours	100	70

Spreadsheet 5. Shows the spreadsheet containing the data on the different resources and their maximum allocation per release.

Stakeholders		
Stakeholder E-mail	Stakeholder Name	Weight
bon_adriel@hotmail.com	Aseniero	9
davidledo89@gmail.com	Ledo	9

Spreadsheet 6. Shows the spreadsheet containing the data on stakeholder weights.

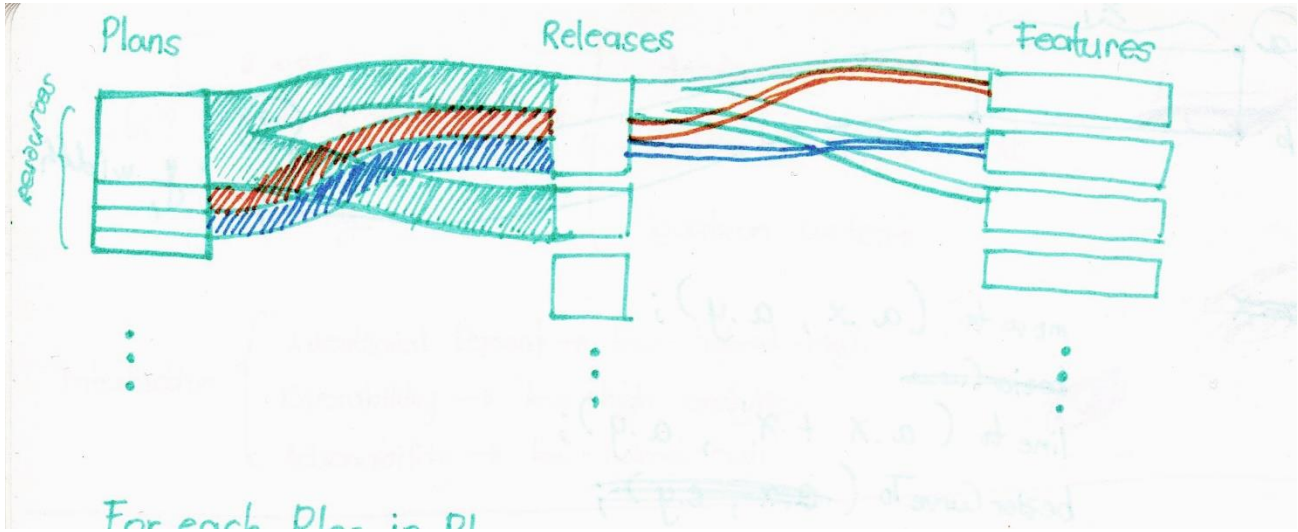
Stakeholder Satisfaction					
	Alternative 1	Alternative 2	Alternative 3	Alternative 4	Alternative 5
Very Excited	3	3	3	3	3
Excited	7	7	6	8	6
Neutral	11	11	10	10	9
Disappointed	5	3	3	4	5
Very Disappointed	0	1	1	0	0
Surprised	1	2	4	2	4
Very Surprised	0	0	0	0	0

Spreadsheet 7. Shows the spreadsheet containing the different levels of stakeholder satisfaction for each alternative plan.

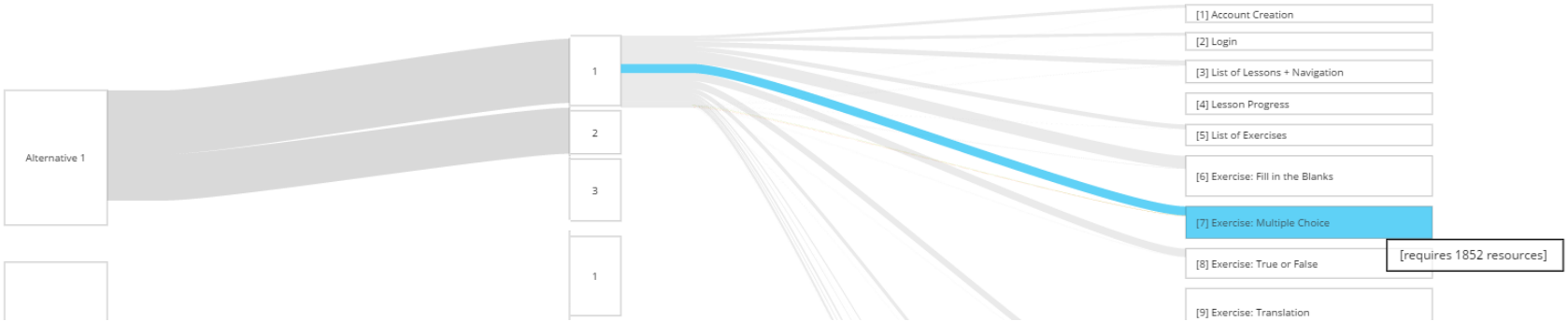
Appendix II

PREVIOUS ITERATIONS OF STRATOS

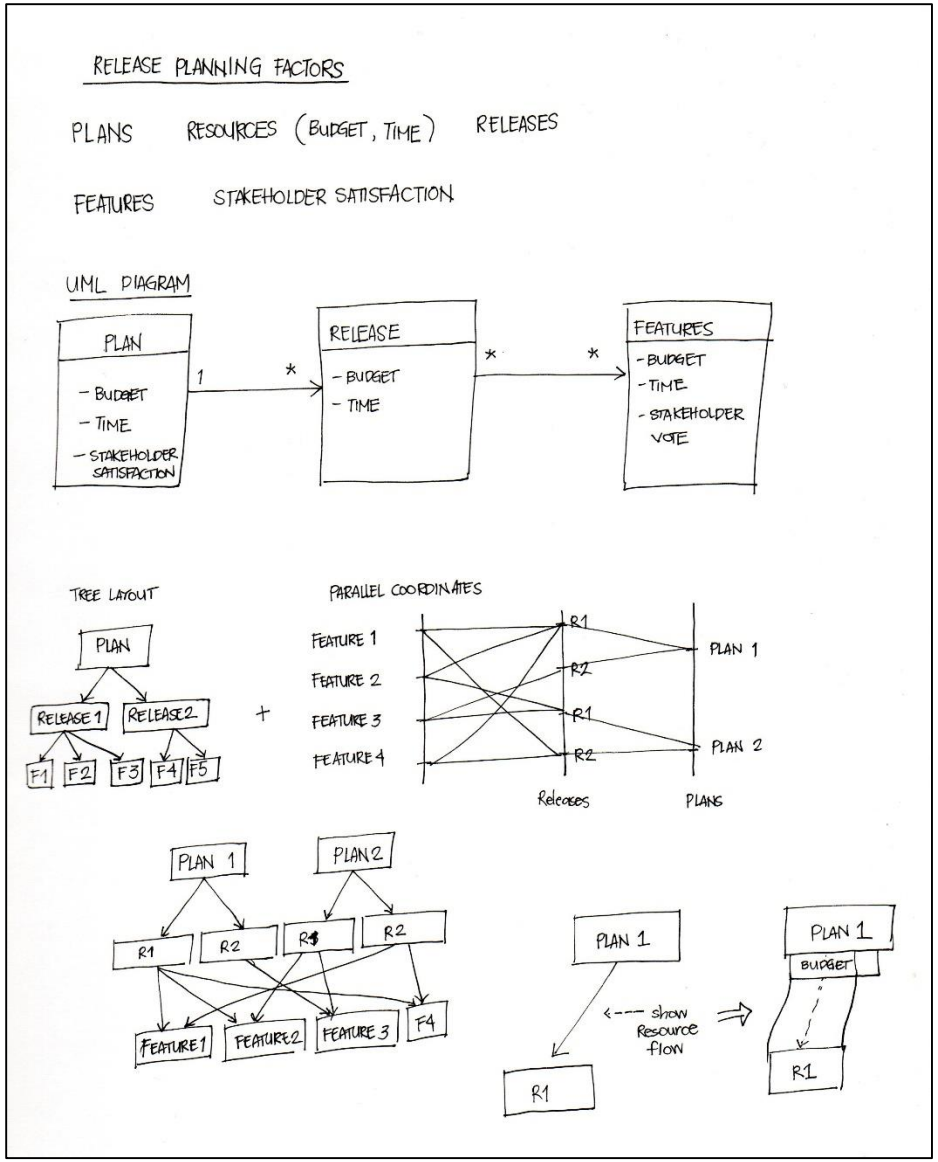
Appendix II contains previous iterations of STRATOS from its initial sketches, previous rough implementations, the one used during the study, and its current look and feel.



Sketch 1. The first sketch concept of the hybrid visualization of STRATOS. The initial design was to have a horizontal flow layout from left to right. This design was discarded later on in favour of the tree layout which emphasized the hierarchical nature of the data more.

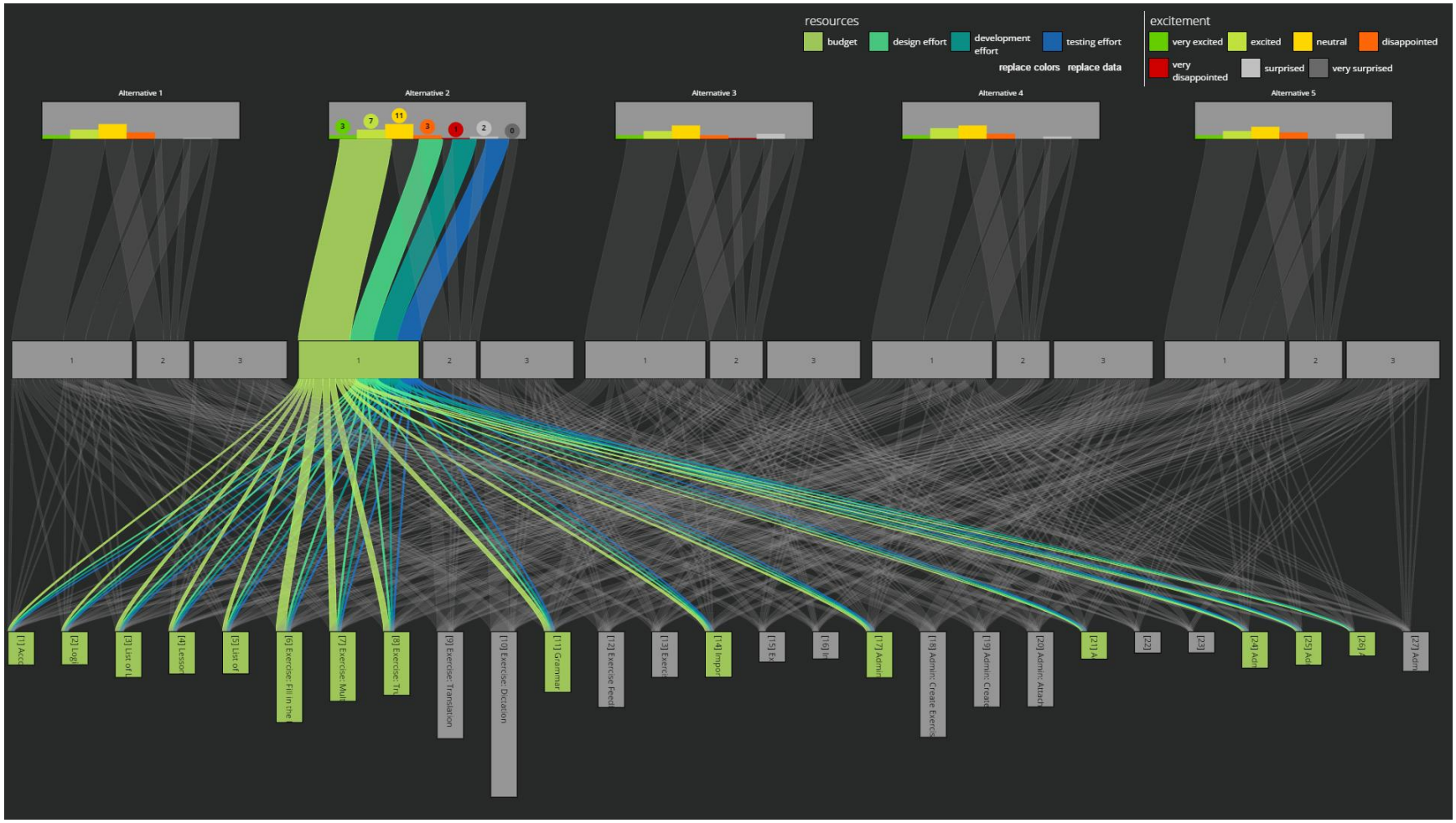


Iteration 1. The first rapid prototype of STRATOS based on Sketch 1.

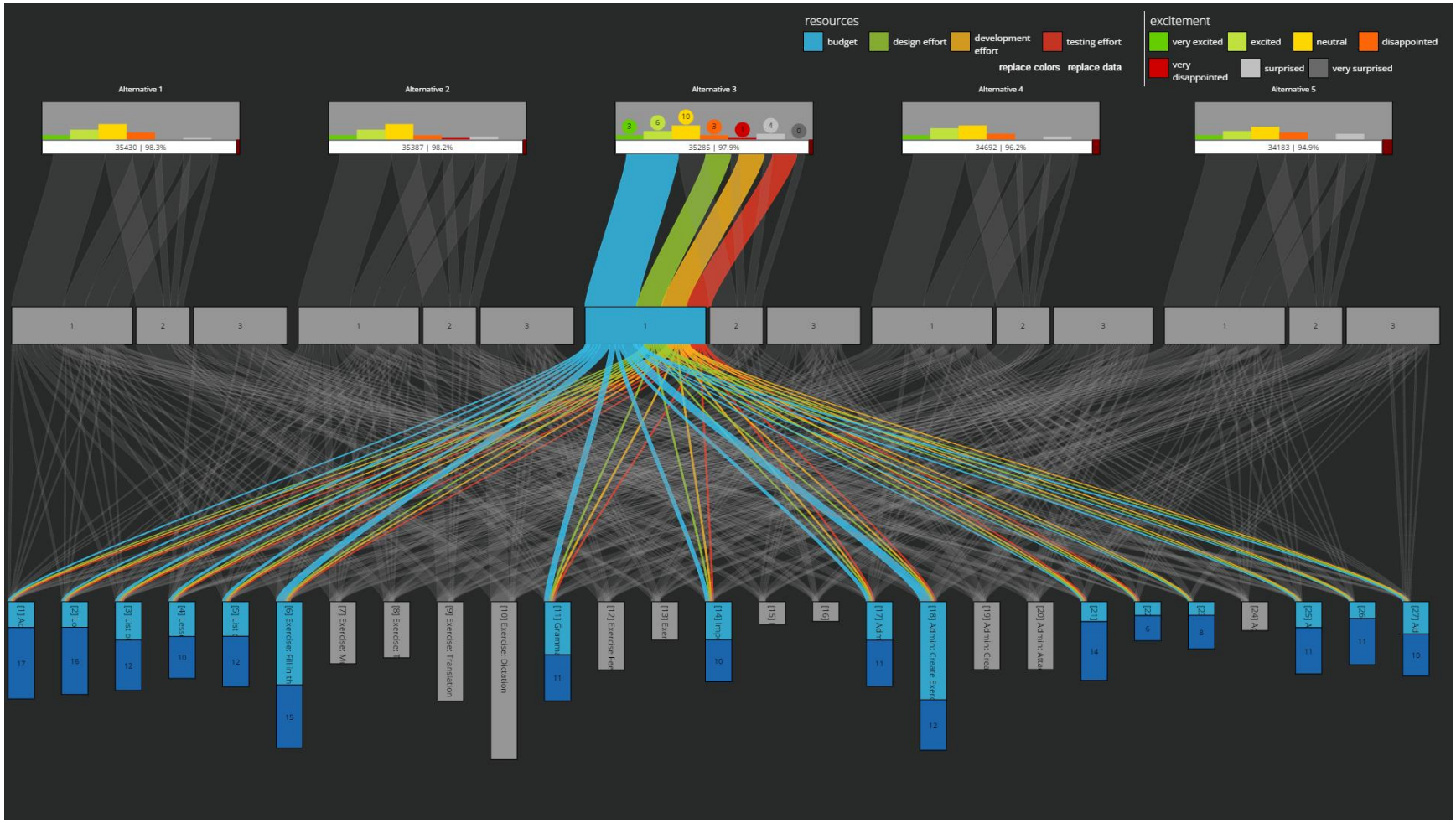


Sketch 2. A sketch revisiting the structure and layout of STRATOS. In this iteration, the tree layout was added.

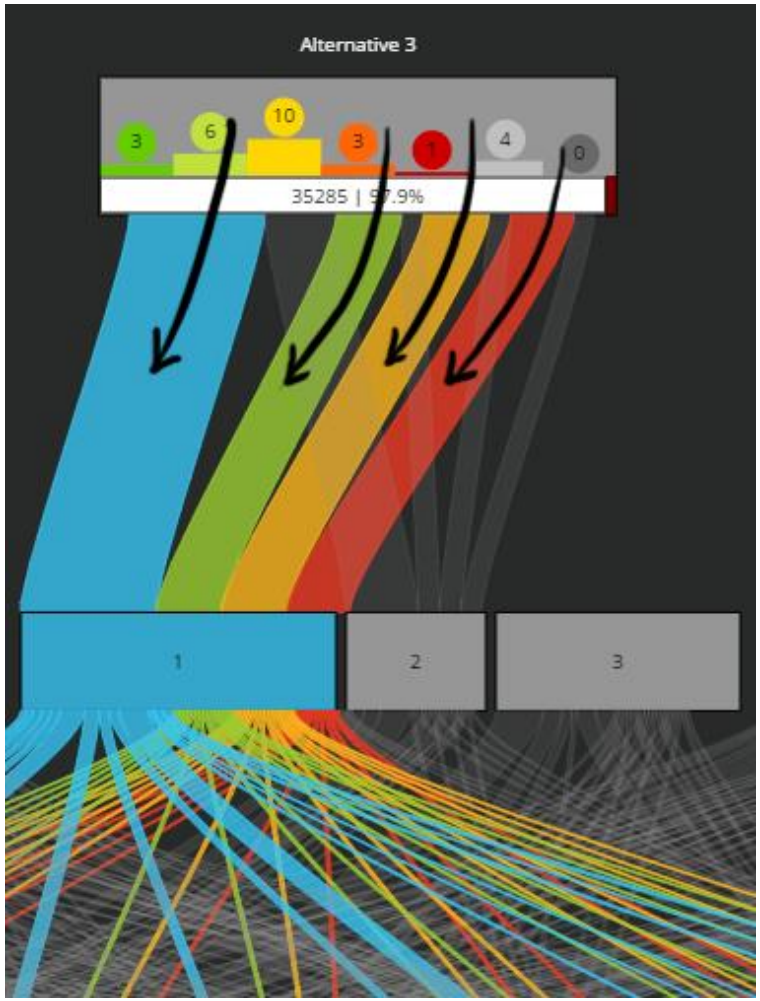
UML diagram sketches helped in the conceptualization of the different visual elements.

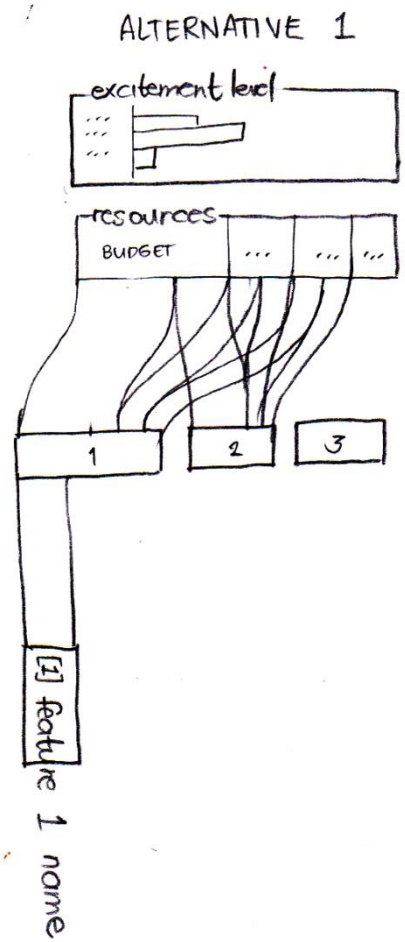


Iteration 2. A version of STRATOS that do not visualize information on feature priority votes and stakeholder feature points. The stakeholder satisfaction bar charts were also previously drawn vertically.

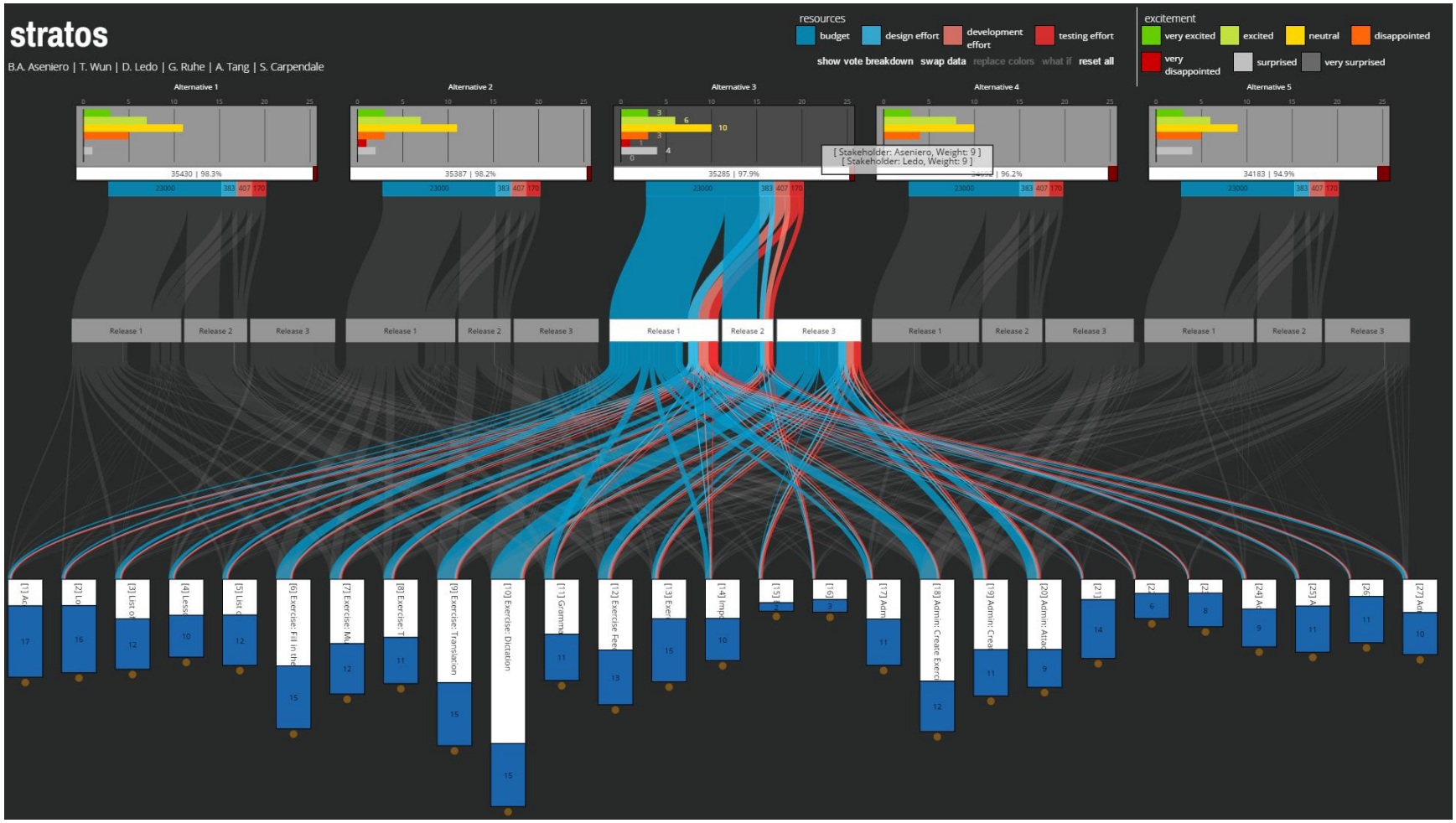


Iteration 3. After showing the visualization to the domain expert, the visualization was improved. This iteration included stakeholder priority votes on each feature, and the stakeholder feature points bar (drawn under the stakeholder satisfaction bar chart).

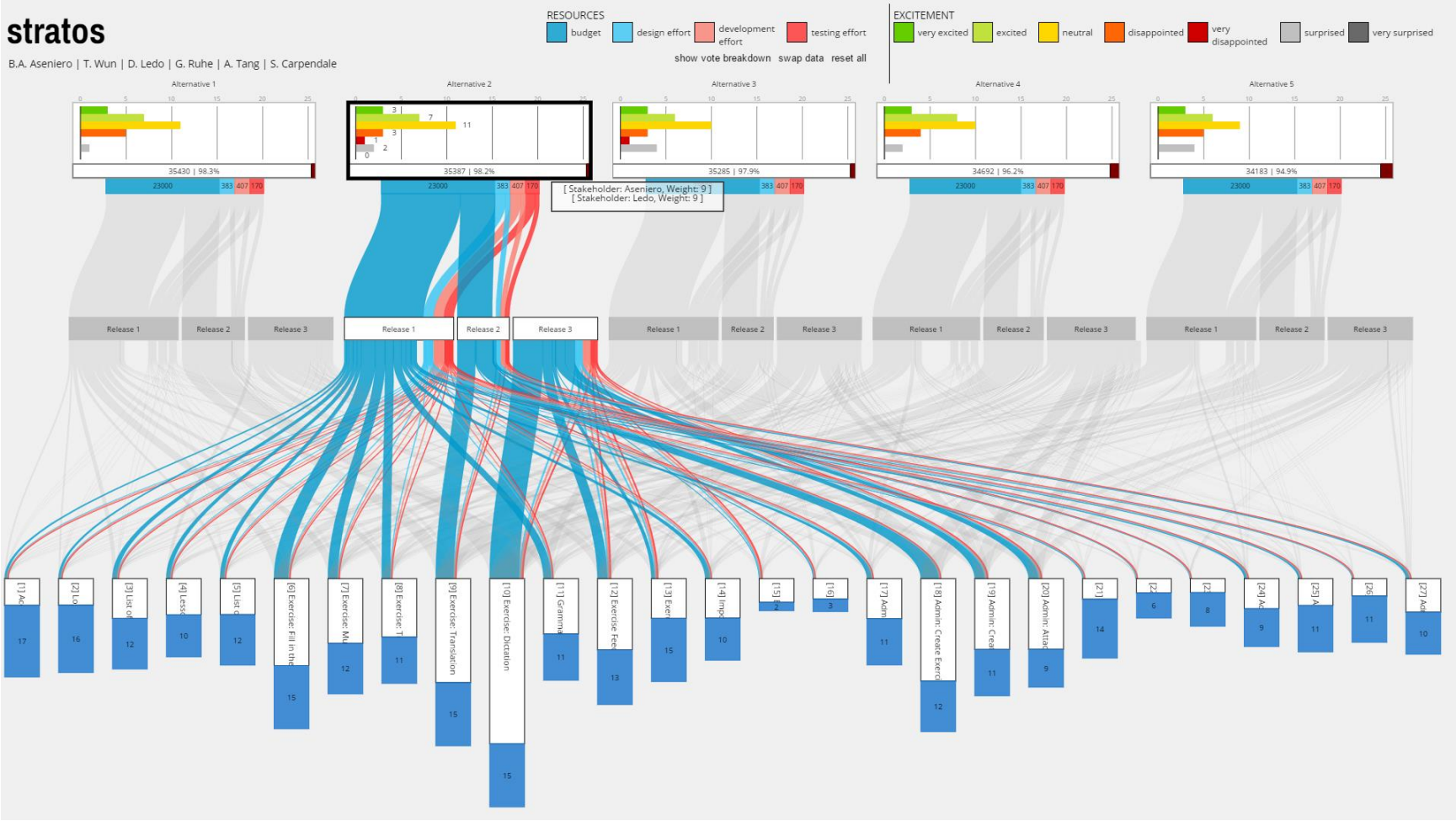




Sketch 3. After a pilot study, some changes were made to the visualization. The previously vertical bar chart representing stakeholder satisfaction was redrawn horizontally because the participants during the pilot kept associating them with the flow of resources (i.e. they thought the bars of the chart were the resources that flow down into the releases rather than stakeholder excitement levels). The bar representing the initial allocation of resources was also added to further disassociate the stakeholder satisfaction bar charts with the flow of resources.



Iteration 4. The look and feel of STRATOS during the study. This iteration is the same with the one presented in this thesis with the only difference on colour palette.



Iteration 5. The final iteration of STRATOS presented in this thesis showing the solution set visualized during the exploration phase of the study.