MASTER THESIS
in
Universal Design of ICT

May 2016

Designing the User Interface of a Technical Application

Mathiesen, Frode

Department of Computer Science

Faculty of Technology, Art and Design

OSLO AND AKERSHUS
UNIVERSITY COLLEGE
OF APPLIED SCIENCES

1 Preface

This report is the final delivery in the master program Universal Design of ICT. The study was completed with the assistance of Schlumberger, who initially proposed the project with the aim of obtaining new ideas and suggestions for improvements to the graphical user interface of OLGA. The project was selected on basis of new challenges and the ability to gain more knowledge and experience with usability and design.

The project has been challenging in multiple aspects such as essentially being a novice in both design and usability further complicated by the broadness of usability, and the unfamiliar domain flow assurance.

I want to convey my gratitude, to all the participants who have provided information and feedback. I would also like to thank my supportive advisors Kato Griff Klæboe, Tor Sommersel and Federico Sporleder from Schlumberger and Anis Yazidi from HiOA.

2 Summary

This study evaluates and suggests improvements that are focused on usability, for the user interface of OLGA. OLGA is a simulator that is used in flow assurance and is a user interface and a simulator that allows users to build and define models of production systems, and simulate the transport of multiphase fluids within that system, including the visualization of the simulation results in line charts.

Usability is a broad concept that has multiple perspectives and different approaches to the operationalizations of usability. Additionally, a majority of current usability literature and studies are mainly focused on mainstream users and interfaces for web-applications and systems that rarely require domain specific knowledge. We learn that further development of OLGA, should focus on improvements focused on stability and efficiency of the interface and bundled tools. Additional effort should be focused on updating and improving the documentation to assist in filling gaps in knowledge, for novices and experienced users. Furthermore, integrated plotting functionality, which has not been updated in recent versions, may benefit of a more focused effort to fix bugs, improvements to completing common tasks, and replacing the current configuration window with a task-oriented interface. Additional improvements, includes increasing the accessibility to plots and provide functionality for exporting data.

In this report, we discuss relevant concepts, such as, usability, design and user-centered design, including different perspectives, guidelines and methodologies. We present and discuss current literature that is focused on usability design for technical and domain-specific systems. A detailed presentation and discussion about the graphical user interface of OLGA is provided, followed by the results from individual interviews, focus groups and the evaluator’s observations and experiences when using the system, including aspects that are relevant for universal design. Additionally we present suggestions and concept designs based on the gathered information. Finally, we discuss the project and present factors that can be used to operationalize usability specifically for OLGA.

3 Table of Contents

3.1 List of Figures

3.2 List of Tables

4 Introduction

Flow assurance is a domain focused on ensuring the continuous and economical transport flow of multiphase fluids (oil, gas, water) in production systems. The system (OLGA) is a simulator used to simulate and predict the behavior of multiphase fluids within pipelines, in various production environments. The system includes functionality to build and define models, define conditions for simulations and functionality for visualizing the simulation results in line charts.

This project focus on the user interface (UI) and is intended to be a precursor to further research and development of OLGA and its interface. A focus of this report is to discover and highlight *areas* that require more investigation, as well as to propose possible improvements to interface elements or behavior of the system. The approach used in the project was, essentially, that of a novice in usability, flow assurance, OLGA and design. The concepts of usability and design were known, *superficially*, and mainly in context of less technical systems with a mainstream user base. Due to the broadness of usability, the project focused on acquiring a deeper understanding of usability and methods to apply usability in design, and generated the question: *In the context of OLGA and its users, what is usability?* This question was formulated due to the majority of current literature about user interfaces primarily focus on web-applications and interfaces for systems that rarely require technical domain knowledge.

Regarding the interface, a discovery was that the further development of OLGA should focus on improving *stability, efficiency, documentation, exporting data* and *plotting functionality*. The project operated with loose boundaries, with some more focus on plotting functionality due to this remaining unmodified in most versions of OLGA. The bundled tools are mentioned in the report, but were considered external functionality and not part of the focus, due to lack of access. The guiding questions during the project were:

- In the context of OLGA and its users, what is usability?
- Which usability principles, guidelines and attributes are relevant for OLGA?
- Which aspects of the interface require more focus in further development?
- Which interface elements may require improvements?

5 Concepts

In this chapter, the various concepts investigated during the span of the project are introduced, including usability, design, and user-centered design (UCD), with a more detailed introduction of usability. An additional introduction to Microsoft Windows Design Principles and Guidelines is also included.

5.1 Usability

The concept *usability* is often summarized as the ease-of-use of a system. The concept was introduced in the early 1980s, with the purpose to replace the term "*user friendly*" (McNamara & Kirakowski, 2005). According to Bevan, Kirakowski, and Maissel (1991) the term user friendly had acquired a host of vague and subjective connotations, the authors further speculated that the usability concept had suffered the same fate. Both concepts are similar in that they aim to make systems and interfaces easier to use, however, usability further aims to match the system closer to the needs and requirements of the users.

Usability is frequently referred to as a *product property*, summarized as *ease-of-use* and often only considered in terms of the UI (Bevan, 1995; Mayhew, 1999). However, the term is not limited to only software and includes additional aspects, such as, hardware and anything humans interact with (Gupta, Ahlawat, & Sagar, 2014).

Over the years, usability has become an important part of human-computer interaction (HCI) and UCD (McNamara & Kirakowski, 2005). Despite the importance of usability in the fields of HCI and UCD, there is no consensus on a definition of usability among researchers and practitioners (Gupta et al., 2014), resulting in a concept that is difficult to implement (Seffah, Donyaee, Kline, & Padda, 2006). Factors that make it difficult to conceptualize, and in turn reach consensus on usability, is that the usability of system depends on the context in which the system is used, who the users are, and if the system is appropriate to the context (Brooke, 1996) (*i.e.* is the user experienced or a novice? Where is the system used and why is it used? Is the system suitable for the intended use?). The dependence on context, users and appropriateness influences the difficulty in conceptualizing usability, what works for one type of system may not be suited to another type of system. Furthermore, it is easy to discover problems due to lack of usability, though users may not label the problems as a usability problem, but blame their own experience or knowledge (Chou, 2002).

It is also important to note that usability is not absolute, the usability of a system may evolve over time, as users become more familiar and experienced with the system (Bevan, 2009). Furthermore, one user group may evaluate and experience a system as having good usability, while another user group may evaluate and experience a system as having poor usability. For example, more experienced users may be familiar with the system and terminology, have developed habits that solves tasks subconsciously, and received training that may shape the perception and experience with the system. In contrast, less experienced and novice users may not have developed habits, be unfamiliar with terminology and may not understand why the system behaves the way it does, which may shape the perception and experience negatively. The situations described can also be viewed in the reverse, if an existing system is redesigned to be easier for novices, it may result in difficulties for more experienced users who rely on previous experiences and habits.

5.1.1 Perspectives

Usability has been around for a long time, with multiple competing perspectives formulated over the years. The varying perspectives take different approaches to conceptualize and define usability, including how to measure usability.

Bevan et al. (1991) listed three perspectives that affect the methods used to measure usability. The perspectives are *product-oriented*, *user-oriented* and *user-performance*. The product-oriented perspective is synonymous with the product-property perspective described below. The user-oriented perspective holds that the usability of a system can be measured in terms of the exerted effort and attitude of the user. The user-performance perspective holds that the usability of a system, can be measured by examining the users' interaction with the system, including a particular focus on how easy it is to use the system, or if the system will be used in the real world.

Three of the approaches to usability are usability as a product-property, quality of experience and quality of use (McNamara & Kirakowski, 2005). Additionally, some experts, such as Nielsen (2012), considers usability as a quality attribute of the UI, used to assess how easy the interface is to use.

With usability as a product-property, the usability of a system is influenced by the presence or absence of features and ergonomic (Bevan et al., 1991). With the approach, usability goals can be realized by adding new features. In the early stages, usability as a product-property was influential, it is still in use but was later replaced, as it was considered too static and situation specific (McNamara & Kirakowski, 2005).

The quality of experience perspective is incomplete and have three main approaches that is being investigated (McNamara & Kirakowski, 2005). The first alternative involves revising the how usability is operationalized and aiming to include more attributes that are subjective. Moreover, it proposes that usability is split into *behavioral* and *emotional* usability. Behavioral usability is focused on the ability to complete functional and goal-oriented tasks with efficiency, while emotional usability is focused on the desirability or need for the system (Logan, 1994). The second alternative proposes new concepts that are hypothesized to be important to consider, but different from usability (McNamara & Kirakowski, 2005). The third alternative, believe that the concept of experience, is under-developed and used without a deeper understanding of what experience is. Thus, the third alternative focuses on achieving a deeper theoretical and philosophical understanding of experience (McNamara & Kirakowski, 2005; Wright, McCarthy, & Meekison, 2005).

Quality of use is the perspective that replaced usability as a product-property, and is currently the most influential perspective. Quality of use holds that usability depends on *who* the users are, *where* the system is used and *why* the system is used (Brooke, 1996). In this perspective, usability is abstracted into measurable attributes where a majority of the used operationalizations includes attributes of subjective nature, requiring feedback from users (McNamara & Kirakowski, 2005).

5.1.1.1 Attributes

With over two decades of history, a variety of definitions, methodologies and new discoveries have led to many usability attributes. In their study, Gupta et al. (2014) included a list of attributes gathered from usability experts. In Table 5-1 below, we list some of the main attributes as defined by Gupta et al. (2014).

Attribute	Description
Aesthetics	User’s rating of how good-looking the UI design is
Attractiveness	How attractive the user finds the system in aesthetics, interaction and utility
Consistency	Consistency of UI elements, language and procedures
Ease-of-use	The user’s rating of how easy the system is to use
Efficiency	How much time and effort is required to complete specified tasks and goals
Errors	How many errors users make and how they recover from the errors
Knowability / Affordance	Whether the system provides guidance on how to use it, and the results of actions
Learnability	How fast new users can learn to use the system and accomplish basic tasks
Memorability	How much the user remembers from previous use of the system
Reusability	How much knowledge the users can use from previous experience with similar and other systems
Reliability	How much the user can rely on the system to complete tasks/goals without losing/destroying work
Satisfaction	The user’s subjective satisfaction during and after using the system
Usefulness	Users’ rating of usefulness of a system to complete specified tasks/goals
Utility / Effectiveness	Whether or not the system provides the necessary functionality to complete tasks and goals

Table 5-1: List of usability attributes with descriptions

Other factors, sometimes included as attributes, are documentation and training, which influences users’ understanding and experience with a system, these factors, including other attributes such as Learnability, further influences the measurement of other usability attributes, *i.e.* the amount of training a user have received, may influence how they would rate some attributes.

5.1.2 Standards

The purpose of standards is to provide implementers with a reference to follow, so that technical, safety, regulatory, societal and market needs can be accommodated (IEEE, 2015). An additional function of standards is to impose consistency in planning, process, development, testing and success criteria. International standards have had limited influence, in terms of consistency of interface components where *de facto* industry standards have had the most influence (Bevan, 2006).

Standards are commonly known to specify requirements needed to be fulfilled to comply with the standard. However, HCI and usability standards rarely specify requirements for the interface, but rather provide general principles, that can be used to guide the development of interfaces and procedures. The standards avoid dictating how the UI should be, so that designers have the flexibility to develop new types of interfaces. HCI and usability standards provide authoritative statements of good practices, mainly focusing on the use of the system; UI and interaction; development processes; and organizations capabilities to apply UCD. Due to HCI and usability standards avoiding specific requirements, it can be difficult to measure the conformance with the standards (Bevan, 2001).

5.1.3 Definitions

As previously mentioned, usability does not have a universally accepted definition. In this section, some of the formulated definitions are presented. This is not intended to be a complete overview of all available definitions.

IEEE provided the following definition of usability, which dates to early in the 1990s:

> *"Usability: The ease with which a user can learn, operate, prepare inputs for, and interpret outputs of a system or component" (IEEE, 1990)*

The IEEE definition provides measurable attributes in Learnability and efficiency in preparing inputs and interpreting outputs, *i.e.* interacting with the system and understanding the feedback.

Furthermore, Nielsen (2012) provides a definition of usability as, *"usability is a quality attribute that assesses how easy user interfaces are to use"* and that five quality components define usability. These components are *Learnability*, *Efficiency*, *Memorability*, *Errors* and *Satisfaction*. Nielsen (2012), additionally, notes that usability also refers to methods for improving ease-of-use during the design process.

In addition, the International Organization for Standardization (ISO) has provided multiple definitions of usability. In the ISO/IEC 9126 standard, usability is defined as:

> *"A set of attributes that bear on the effort needed for use and on the individual assessment of such use, by a stated or implied set of users" (ISO/IEC, 2001)*

A main criticism of the later definition is that other than assessing that usability are attributes which have an effect on the effort and individual assessment of use, no measurable attributes are specified. According to Gupta et al. (2014) the standard also redefines the definition, as the capability of the software to be understood by the users under certain conditions.

According to the multi-part standard ISO 9241-11, usability is defined as:

> *"The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" (ISO, 1998)*

The ISO 9241-11 definition is widely referred to in literature, but to which extent it is adopted by practitioners is unknown. The definition provides three measurable attributes, which can be further divided into multiple measurable properties. Moreover, it emphasizes that the users and context must be specified. The definition has received some criticism, Bevan (2009), with a focus on user experience (UX) noted that the definition "have nothing to say about Learnability", which changes usability over time, and further how the definition does not say anything about how user experience evolves from expectations, interaction and reflection on the experience to a total experience. Though the criticism comes from a perspective, mainly, focused on UX it is still relevant, as expectations influence the stated measurable attributes.

Regarding the ISO standards, it must be noted that ISO/IEC 9126 has been replaced with ISO 25010 (ISO & ISQS, 2011). The ISO 9241 series have been subject to a revision, resulting in changes in the numbering of the parts. The change in numbering was to make it possible to cover more topics in one part, where parts with two zeros (ex. 100) indicate a generic or basic standard, parts with one zero (ex. 110) indicate the standard regulate fundamental aspects, and parts with no zeros (ex. 111) regulate specific aspects. With the new number system, the usability definition is located in part 210 (9241-210). An effort has also been made, to use the ISO 9241 usability definition consistently in all standards referring to usability.

5.1.4 Principles

Usability design principles provide guidance on aspects to consider when creating a design. Principles do not dictate how the design should look, but can be used as a reference to guide design. In addition, they can also be used as a form of heuristics. Cognitive sciences is relevant to usability, listed below are seven areas from cognitive sciences as defined by Experience Dynamics (2015), which are covered by the various sets of principles.

1. Perception – Recognition and interpretation
2. Attention/Cognitive overload – Confusing or overwhelming
3. Memory – What is remembered after use
4. Pattern recognition – Recognized patterns in interaction and visuals
5. Mental models – Conceptual model of how the system works
6. Affordances – Discoverable potential action from the object or environment
7. Emotion – Emotions elicited from the use of the system

During the process of gathering literature, it was observed that some authors separated mental models and conceptual models, where a mental model referred to the users' model of how a system works and behaves, while a conceptual model referred to the developers' model of how the system work and behave. In this report, "mental model" and "conceptual model" is used to refer to one concept, the mental model users have, of how the system works and behaves. Some characteristics of a mental model, it is subjective and evolves through interaction, constrained by the users' technical background and experiences. Furthermore, the model is incomplete; it should be functional, but do not need to be accurate (Donald A Norman, 1983).

5.1.4.1 The Eight Golden Rules

Shneiderman, Plaisant, and Cohen (2009) developed the Eight Golden Rules based on their experiences and refinement over two decades.

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogs to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Consistency – Similar situations should require consistent action sequences; terminology should be consistent in menus, prompts and in all text feedback; color, layout, while other screen elements should be consistent in use and look. As a result, the users predict what will happen and consequently we avoid surprises. Exceptions to this principle should be kept to a minimum.

Universal usability – Users are diverse and the design should reflect this fact. Design for plasticity and facilitating transformation of content. Features for novice users such as explanations, and for expert users such as shortcuts and functionality for speeding up interaction with the software should be added.

Feedback – There should be feedback from the system for every action the user performs. For frequent and minor actions the feedback can be modest (for example changing color), for infrequent and major actions the response should provide more information to the user.

Yield closure – Action sequences should be organized in groups with a beginning, middle and an end. Informative feedback at the end of an action-sequence clearly indicates what has been done, and that the task is completed. This is also relevant for single-step actions, if activating a button, which initiates a task where the output is placed in another location, providing feedback that something started and ended is needed.

Prevent errors – The design should, as much as possible, be designed so the user is prevented from performing errors, for example disabling/hiding unnecessary screen elements; prevent invalid data input in input fields. The interface should detect possible errors, and offer simple, constructive and specific instructions for recovery. Preferably, the system state should be unchanged when encountering errors, or instructions on how to recover the previous system state should be available.

Reversal of actions - All actions performed by the user should be easily recoverable. The benefit of this is that the user knows they can undo errors, encouraging the exploration of the system.

Locus of control – Give experienced users the feeling of being in control of the interface, and ensure that the interface respond to their actions. Avoid surprises or changes in system behavior that are familiar to the users. Users are annoyed by tedious data-entry, difficulties obtaining necessary information, and inability to produce the desired results.

Memory load – Humans capacity for processing information in short-term memory is limited, thus the interface should not require that the user remember information from one screen to be used in another.

5.1.4.2 Norman's principles

Donald Norman developed two sets of principles for design in "Psychology of Everyday Things" and in the revision "The Design of Everyday Things". These principles are focused on what the system should be designed to do, or what to consider when creating a design.

1. Visibility
2. Conceptual model
3. Affordance
4. Mappings
5. Constraints
6. Feedback

Visibility – focus on making things visible for the user. The system should let the user know what state it is currently in, and what actions are available for the user. Actionable elements should also be made visible, *i.e.* be visibly different from non-actionable elements. This includes differentiating between important and unimportant aspects, *i.e.* unimportant aspects should not steal the focus from important aspects.

Conceptual model – the interface design should be designed so the conceptual model of the functionality and behavior is consistent for all of its system states. A conceptual model is created in the users' minds and is used to describe how a system works, and to predict how the system will respond to actions.

Affordance – Design so that an object's properties can be identified by looking at it, *i.e.* by looking at the object a user should be able to tell what they can do with it.

Mappings – The interface should include good mappings. Performing an action on a control should produce an expected effect. Perceived groupings should also be used.

Constraints – The interface should include restraints, preventing user error and making it visible for the user what is relevant in the context.

Feedback – The interface should provide feedback on all actions performed by the user, informing that the action has been registered.

Additionally, in an earlier revision Donald A. Norman (1988) provided the design tips described below.

Use knowledge in the world and knowledge in the head – The design should assist the user in creating a conceptual model of how the system works.

Simplify the structure of tasks – The processes that are used in tasks should be made visible. However, the control should not be removed from the user.

Design for error – Regardless of experience, users make mistakes. The interface should be designed to expect errors, and make it easy to reverse actions.

Standardize – The mapping system should be standardized and used consistently if arbitrary mappings are needed. Standardizing enforces consistency, making it easier to develop a conceptual model of the system. This also implies following platform conventions, as well as common standards in the domain where the system will be used.

5.1.4.3 Nielsen's Usability Heuristics

Below we describe a set of usability principles for interaction design as defined by Nielsen (1995), called *10 Usability Heuristics for User Interface Design*. These heuristics can be applied in evaluation of a system, or the development of an interface for a system.

1. Visibility of system status
2. Match between the system and real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognitions rather than recall
7. Flexibility and efficiency in use
8. Aesthetics and minimalist design
9. Help users recognize, diagnose and recover from errors
10. Help and documentation

Visibility of system status – The user should be informed by the system about what is going on, through appropriate feedback within reasonable time.

Match between the system and real world – The system should use language and concepts familiar to the user and relevant to the purpose of the system. The system should follow real-world conventions, information should appear in a logical and natural order.

User control and freedom – Users should have the possibility to undo and redo actions. System functions should have obvious "emergency exits" in case the user wants to leave an unwanted system state.

Consistency and standards – The system should follow platform conventions so the user does not have to wonder if different words, situations, or actions mean the same thing.

Error prevention – An interface design preventing problems from occurring is better than good error messages. Eliminate error-prone conditions or check for errors and present them to the user with a confirmation option before they commit an action.

Recognitions rather than recall – Make objects, actions and options visible to reduce the memory load on the user. The user should not have to remember information from one part of the dialogue to another. Instructions should be visible, or easy to retrieve when needed.

Flexibility and efficiency of use – Implement accelerators, such as shortcuts, not visible to novice users. These can be used by more experienced users to speed up the interaction with the system. This way the system can cater to inexperienced and experienced user, the users should also be able to tailor frequent actions.

Aesthetics and minimalist design – Dialogues should only contain information that is relevant, and needed. Every extra unit of information in a dialogue competes with the relevant information, and renders the useful information less visible.

Help users recognize, diagnose and recover from errors – Error messages should indicate the actual problem, and offer constructive information to solve the error. The messages should not be expressed in only error messages.

Help and documentation – Even if the system can be used without documentation, it may be necessary to provide help and documentation. Such information should be easily available and focus on the user's task, list concrete steps, and not be concise.

5.2 Windows Design Principles and Guidelines

The system (OLGA) discussed in this report is developed for the Microsoft Windows platform. The Microsoft Design Guidelines (MDG) and UX design principles formulated by Microsoft for the Windows platform can be used to guide the interface design. By following the guidelines, the system is developed to be consistent with the standard platform functionality, making it easier to reuse previous knowledge from the platform.

The guidelines and principles provides best practice for layout and designing the system to appear as a native system (developed for the platform), with the result that users familiar with the platform can use existing knowledge of information and functionality is located, or even predict behavior.

5.2.1 UX Design Principles

Reduce concepts to increase confidence – When concepts are added, and for existing concepts, there should be a necessity for the concept. Concepts should have a purpose and be needed, and in the overall experience be consistent.

Small things matter, good and bad – Small details in the UI, tasks and problems are important, the small details are used frequently and small details may result in a majority of inconveniences. Known bugs and inconveniences in the UI and system should be fixed.

Be great at “look” and “do” – If the program is great at a specific task then the UI should reflect this, the UI should make it obvious to the users what they can do with the interface.

Solve distractions, not discoverability – Remove unnecessary information and controls grabbing the user’s focus. Steps should also be taken to prevent features from competing with themselves. Furthermore, a “first run experience” or a tour of the UI will not fix discoverability.

UX before knobs and questions – The number of questions asked by the UI should be reduced. If the UI has numerous dialogs asking for information or parameters then they should be consolidated to ask for the most important information.

Personalization, not customization – With customization the user must configure how the interaction with the program should work, how the layout is, etc. with personalization this is implicit and happens over time, where the UI adjust itself to provide the functionality the user is most likely to use.

Value the life cycle of the experience – The experience with the system begins with the installation and creation of the program through: first use and customization, regular use, management and maintenance, and uninstall or upgrade.

Time matters, build for people on the go – This principle can be interpreted as relevant for systems used on mobile platforms. Another way to look at it is that, users do not want to spend much time to complete tasks, load files or locate things and the design should be focused on making it easy to locate, prepare and execute actions.

5.2.2 Microsoft Design Guidelines

The MDG (Microsoft, 2015) is split into multiple categories, which are further divided into sub-categories. The guidelines cover how to, when to and alternatives to implement elements such as buttons, informative text and such. The defined main categories are:

- Controls – covers UI elements users can interact with
- Commands – covers actions users can take while using the system
- Text – covers any text presented to the users, with guidelines for UI text and style and tone
- Messages – covers any kind of message the users may want or need
- Interaction – focuses on optimizing interaction for touch input, but also include other input devices
- Windows – covers how to use the various types of windows and dialogs
- Visuals – covers visual elements, which are not controls
- Experiences – covers common experiences and use cases
- Windows Environment – covers how the environment and features provided by Microsoft Windows can be used

Some categories are divided into multiple sub-categories, making covering the entire MDG time consuming. A few of the controls described in the MDG is introduced below, where checkboxes and tree views are used in OLGA, while balloons is an element that could be included in the interface.

Balloons are small pop-up windows that can be used to inform users of non-critical problems or special conditions in a control. With balloons the source of the message is identified with a tail, the balloon may include an icon, a title and a text body. For example, if a control automatically fixes an error in input, a balloon can be used to inform the user that something happened, why it happened and how the user may rectify the fix.

Checkboxes allow users to make decisions between two opposite choices (*e.g.* enabled | disabled), and should be used to toggle options on or off or to select or deselect items. In situations where the selected and unselected states are not clear opposites, or the meaning of the cleared state is unclear another control, such as radio buttons, should be considered. The phrasing used should also be positive, selecting a checkbox should not mean that actions should not be performed.

Tree views enable interaction with hierarchically arranged objects, and display relations between objects. Tree views should be used when a hierarchy of more than two levels, excluding the root node, needs to be displayed.

5.3 Design

The word design has many applications; design may refer to the look and feel of a product or interface, or to how a process is structured among other things. In this report, design refers to the GUI’s look and feel.

When redesigning there a few alternative approaches available, such as, identifying tasks performed in the interface and add or redesign the necessary interface elements to improve the action sequences required to complete the tasks. A risk with this approach is that inconsistencies can be introduced to the interface or the actual problem may be ignored. Alternatively, the redesign can focus on a larger part of the interface or a part of the system, and add, modify or replace existing interface elements. Introducing inconsistencies is still a risk and may break expected behavior. However, when a larger part is redesigned it is easier to create a consistent design for that part. Another alternative is a full redesign, where the whole interface is subjected to a redesign process. A full redesign may be a lengthy process, and risks breaking learned behavior, may require re-educating the users and may not be positively received by the users. However, with a full redesign, inconsistencies can be fixed early and can be used to modernize the interface.

A benefit of conducting a redesign is that the system and most problems are already “solved”, this way the new design can reuse aspects of the existing interface and fix the broken parts. Actual use of the existing interface may also reveal new aspects or workflows that are more efficient or more secure. Users can easily be involved in all stages of the redesign process, dependent on availability of users. In exploratory phases, users can be interviewed, and observer, to identify how they use the software, in the development of the design the users can provide feedback, be included as “co-designers” or in testing to evaluate and test design prototypes. It is not required to include users in the design process, but it is highly recommended, and expected in UCD, as the target users are considered the experts in the domain and are ultimately the ones who will use the design.

Performing a redesign is a time consuming, costly and frustrating process (Johnson, Johnson, & Zhang, 2005), where successful redesigns are partly reliant on complete and accurate evaluations of the existing system (Rose, Shneiderman, & Plaisant, 1995), other factors include knowing the typical environment, knowing who the users are and understanding the users’ goals. Due to the various factors, redesign is closely integrated with analysis that is performed on the existing system and users. The different stages in a redesign process require different skill sets (Richardson, 2013). Gathering data and performing analysis on the gathered data require the skill set of a researcher, who is able to extract the users’ *wants* and *needs* based on observation and questioning. While designing an interface, requires the skill set of a designer, who is able to transform the requirements, and users’ wants and needs into an interface design.

- Preparation
- Data gathering and analyses
- Reporting
- Planning
- Designing

In the preparation stage, the focus is on preparing for the data gathering and analyses. The designers should familiarize themselves with the current system as novice users. Learning the new system includes reading the documentation, use the training material and using the system and if possible attend training sessions (Rose et al., 1995). Setting initial goals and preparing questions is also important in the preparation stage, which is later used during observations or interviews. During data gathering field studies may be conducted, evaluations of the existing system is performed. The data is later analyzed and used to prepare a report of the findings. Findings from the field studies and evaluations may be reported to the stakeholders, and are used to create plans for the redesign. Performing analysis includes multiple data sources and depends on the practitioners' knowledge and expertise (Følstad, Law, & Hornbæk, 2012). According to Følstad et al. (2012) introductory texts to analysis only provide high-level descriptions of usability evaluation, lacking in details on how to perform usability testing and heuristic evaluation. Thus creating a situation, where new practitioners may be required to perform trial and error. There are frameworks for conducting such analyses such as SUPEX (Cockton & Lavery, 1999), however, knowledge and experience is required to locate, as well as, successfully implementing such frameworks in the process.

During the design stage, a new design is created using the plans and findings from the data gathering to make design decisions. Testing can be conducted during the design process, as this will enable quick testing of certain design decisions and give more data that improve the decision-making. More than making the interface look good, GUI design is about communication and behavior. The design should inform the user of what they can do, how to interact with it and what to expect (Wilding, 1998). A system may have a large amount of functionality that requires more GUI elements that increases the complexity and negatively influence the communication (Dickinson, Eisma, & Gregor, 2002; Wilding, 1998). In the process of design, the results from the analyses should be used to inform the decision-making. In essence the process attempts to conform to defined criterions, and principles by making decisions regarding trade-offs (Wilding, 1998), *i.e.* should a functionality be added to the interface at the cost of complexity or should it be placed in a context menu at the cost of discoverability? The goal is to create a design that allows the user to complete tasks, with the optimal trade-offs. What an optimal trade-off is depends on the design goals.

5.4 User-Centered Design

Due to usability relying on knowing who the users are and how the system is used, renders UCD relevant. UCD is a design, and analysis, method based on involving users actively in the design process. UCD seeks answers to questions about the goals and tasks performed by the users. The information obtained from these methods is used to assist in decision-making about the design and development of the interface. The type of questions depends on the type of system, interface and domain. The questions may include:

How much experience do the users have?

What are the users' goals?

What do the users want?

What do the users need?

What is the purpose of the system?

How is the system used?

What tasks are performed?

Is the system missing functionality necessary to complete a goal or task?

What are the frequent actions/tasks/procedures?

What should be improved? (Moreover, what constitute an improvement?)

There are many methodologies suited for UCD available, aimed at answering questions, however, few of these address the methods required in a redesign process (Johnson et al., 2005). In a survey, performed by Mao, Vredenburg, Smith, and Carey (2005), experienced UCD practitioners were questioned on UCD methods applied in the design process. The survey resulted in a list of thirteen methods, used in design processes, and was ranked according to importance and how often the method had been mentioned. Below, some of the methods from the survey are described, in no particular order. The way the methods are applied in a redesign process is up to the practitioner, and relies on the knowledge and experience of practitioners.

Field studies – The designer is immersed in the environment of their users. Allowing them to observe how the users behave in the environment where the system is used, and observe details, which would not be observable in an experiment, or interview (Spool, 2007).

User requirement analysis – A process in which information about the users are gathered, and analyzed in order to establish and document the requirements of the user so that they can be implemented in the process of designing the system (Maguire & Bevan, 2002).

Iterative design – A design process where design prototypes are completed, then tested, followed by fixing the identified issues, followed by iterations of testing and fixing the design (Nielsen, 1993).

Usability evaluation – A process where evaluations are accomplished by identifying representative users, tasks, and developing a procedure to capture the difficulties users have in trying to use a particular software to accomplish the tasks (Scholtz, 2004).

Task analysis – Analyses what actions and/or cognitive process users are required to do in order to achieve a task. A task analysis "provides knowledge of the tasks that the user wishes to perform" ("Task analysis," 2006).

Focus groups – Typically 7-10 individuals, selected based on certain characteristics in common that relate to the topic of the focus group. The participants are usually unfamiliar with each other. A focus group can tell us how *groups of people think or feel* about a particular topic; insight into *why certain opinions are held*; improve *planning and design of new programs*; provide means of *evaluating existing programs* (Marczak & Sewell, n.d).

Formal heuristic evaluation – A method in which one or more reviewers, compare a product to a list of design principles, referred to as heuristics, and identify if and where the product does not follow the principles ("Heuristic Evaluation," 2007).

User interviews – A method used to discover facts and opinions held by users of the system, interviews are usually held one-on-one. Interview reports must be carefully analyzed and targeted to ensure that they make an impact ("Interviews," 2006).

5.4.1 Contextual Design

Contextual design includes techniques, assisting in creating a coherent understanding of how the users work, and enable the designer to create a design from that understanding. To achieve this goal the process consists of explicit steps and deliverables, from initial discovery through system specification. The process is split in multiple parts, covering different stages of the process. The different stages use diagrams to describe the data, as these are easier to share and interpret. The stages are:

- Contextual Inquiry
- Interpretation
- Data Consolidation
- Visioning
- User Environment Design
- Prototyping

Furthermore, McDonald, Monahan, and Cockton (2006) successfully applied the contextual design process as a field evaluation method.

Contextual inquiry is a field interview, data-gathering technique, used to study a few selected individuals in depth. The aim is to obtain a greater understanding of the work practices of the users, by revealing commonalities through inquiry and interpretation. Inquiries are made to obtain a better understanding of the needs and desires of the users, as well as, how they approach work. Usually one-on-one interviews are held with the user in their work environment. By keeping the users in the context, more aspects of how they work are revealed, such as aspects that are unconscious, and aspects difficult to explain. The interview method is designed to address how to get data about the structure of work practice; how to make unarticulated knowledge about work explicit and understandable for designer; and how to get low-level details on the workflow that is habitual and invisible to the user.

Contextual inquiry is based on a set of principles (described below) used to guide the process of collecting data. By basing the technique on principles it allows for adaptation to the situations a design project encounters.

Context – Users are observed and interviewed at their workplace, with the goal of obtaining information only available in the context of use.

Partnership – Users are engaged, with the aim of making the user and interviewer collaborates in uncovering and understanding the users' actions.

Interpretation – Developing a shared understanding between the user and interviewer about the aspects of the workflow that matters and why it matters for the design.

Focus – Before the interview, the purpose and the focus of the interview must be defined. The interviewer needs to guide the user in talking about the part of the work that is relevant to the design.

The most basic form of contextual inquiry is a contextual interview, typically lasting 2 to 3 hours. About 10 to 20 interviews are considered sufficient, where participants have different roles and use the system differently, as little new information is likely to be discovered (Beyer & Holtzblatt, 1997).

During the interpretation stage, the data is analyzed in interpretation sessions. The aim of interpretation is to provide a concrete representation of the work of each user. Work tasks, and details of the working environment are modeled using five models.

Flow model – Represents division of work and responsibilities and how individuals react.

Sequence model – Represents the user's steps to complete a task.

Cultural model – Represents cultural aspects of the work environment.

Artifact model – Represents objects created during the work or used to support the work.

Physical model – Represents the physical work environment.

During the data consolidation stage, the results from the interpretation are consolidated and analyzed to reveal patterns and underlying structure of the work. The result of shows what is important in the work and can be used to guide the design. Consolidation can be achieved by using an affinity diagram, where individual points from interpretation sessions are brought together into a wall-sized hierarchical diagram.

During visioning, the consolidated data is used to identify key issues and opportunities. Discussions focused on how the issues and work can be improved and requirements are held. The visioning stage is focused on creating a vision for how the new design will affect the work. Storyboards are used to develop the vision into a definition of how the new system will be used.

The user environment design (UED) can be seen as a floor plan for the new system, where the different parts of the system are shown. Each part also includes the functionality available and how they relate based on the viewpoint of the user. An explicit UED supports making sure the structure is right for the users, and can be used to plan integration and project management across engineering teams.

In the prototyping stage, mockups of ideas are created, and paper prototypes are used to test the ideas. Paper prototypes are recommended, as they are cheap and support rapid continuous iterations of the design. When testing the prototypes new redesigns are created in collaboration with the users, and the result of several prototyping sessions are used to drive the detailed UI design.

5.4.2 Co-Creation

Co-creation, also referred to as co-design, is a methodology based on participatory design. Participatory design has roots from Scandinavian work trade unions, and action and socio-technical design ("Participatory Design," 2005). Participatory design is viewed as an approach where users are involved in the process and studies the unconscious and normally hidden aspects of human behavior that is developed, and used by users of technology (Spinuzzi, 2005).

Practitioners of co-creation, define the concept differently, at its core however, co-creation is about creating something new together with the user. Sanders and Simons (2009) define co-creation as "*any act of collective creativity that is experienced jointly by two or more people*", an alternative definition, defines co-creation as "*an active creative and social process based on collaboration between producers and users that is initiated by the firm to generate value for customers*" (Roser, Samson, Humphreys, & Cruz-Valdivieso, 2009).

In a classical design process, the roles of the involved parties are predetermined, the researcher studies the user, the user is a passive object of study, and the designer receives a report of findings from the researcher. Both the user and designer are passive, when it comes to identifying requirements and relevant knowledge (Sanders & Stappers, 2008). However, with co-creation, the user is actively engaged in the process and contributes to the development of ideas, evaluate ideas and refine ideas and concepts (Vision Critical, 2015). The role of the researcher, is primarily facilitating the creativity of the users by leading and guiding, as well as, providing an arena and a clean slate for creativity. The user is the expert of their experience providing feedback to inform and develop the design with provided creativity tools. The designer and researcher collaborate in the development of the creativity tools given to the user. The researcher and designer can also be one person Sanders and Stappers (2008).

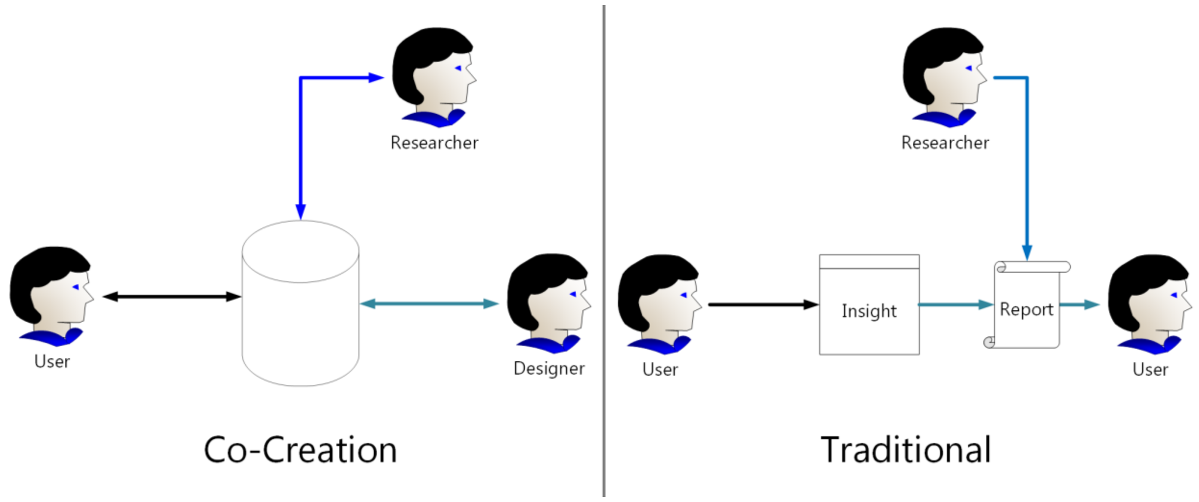


Figure 5-1: comparison between co-creation and traditional design

6 Literature

In the flow assurance domain, there are a few proprietary systems available. Some systems similar to OLGA include:

- PIPESIM – steady-state multiphase simulator (Schlumberger)
- LEDAFlow – dynamic multiphase simulator (Kongsberg Gruppen)
- FloWax – multiphase pipeline simulator (KBC Advanced Technology)
- FlowManager – real-time metering and flow analysis system (FMC Technologies)

Literature specifically targeting some of these systems exists, however, very few studies focus on improving the usability or accessibility of these systems. A majority of the studies are primarily focused on technical optimizations, such as improving the software and physics models used in the systems. Other literature also focuses on the use of the systems to predict behavior in production systems. A lot of the literature includes the keywords usability, however, the term is used to illuminate that a certain formulae or model is "usable" in real-world scenarios. Literature targeting interface, usability and accessibility design of interfaces for flow assurance systems has proven difficult to locate and may not exists.

What is known is that the flow assurance domain is highly technical and complex, similar to other engineering branches, and similar to CAM-software, require domain-specific knowledge and some software tool expertise in order to create something "usable" for a real scenario (Arning, Himmel, & Ziefle, 2016). The required domain-specific knowledge acts as a barrier to novice users, who may be able to use the interface but struggle to apply their knowledge into the interaction. In a study, Arning et al. (2016) discovered that there was a difference in how older and younger novices (older/younger users with low CAM experience) evaluated the usability of their CAM software. Their study illuminated that not only the GUI should be improved but also knowledge support. The areas suggested to focus on in this study were, preventing errors, improve feedback, responsiveness, and improve search functions with context-sensitive search and presentation of results. Recommendations for GUI improvements were centered on behavior and controllability, while elements such as colors, font sizes and other *cosmetics* had low priority (due to the evaluation results).

Despite the systems being complex they are not necessarily, difficult to “use”, abstractions are used to model the production systems, reducing the complexity of engineering concepts (Villberg, 2007). With technical and complex domain specific systems, the challenge is not limited to making users able to interact and use the system, but includes assisting in combining and matching the users’ knowledge with the system, to create an end-result that is usable.

The proprietary nature of the available flow assurance systems may make it more difficult to perform comparative analyses of the interfaces. Comparative studies, such as, StreamLine (n.d) that compares the simulation results exists, however, studies that compare the usability and accessibility of the systems does not exist.

Other complex software include CAD, CAM and CFD, these are used for other engineering branches, which also require domain-specific knowledge. Similar to the flow assurance domain, the literature are primarily focused on technical optimizations. However, some studies focus on development of GUIs to assist users in preparing models. One such study, performed by Bhasin and Venkata (2009), developed a cross-platform framework and a GUI to assist CFD users. Mainly the GUIs construction was described, and list “usability” as a feature of the interface without further specification. It must be noted though that such studies improve the usability and access by replacing textual descriptions of models with visual representations. Such studies primarily focus on a single aspect of the modeling process, which make it possible to reduce complexity and keep things simple.

The goals of using GUIs in these domains are to reduce the complexity and increase the access to the domain. GUIs provide some benefits such as removing the need users’ not need to learn syntax for command-line systems, as well as making it easier for novice users to use the system. On top of keeping track of models, usually of large and complex systems, users are constrained by the human ability to handle complexity and shortcomings of the “working memory” (Villberg, 2007). Novice users may be discouraged and overwhelmed by facing the task of modeling a large complex system textually, but may be encouraged by the simplicity of modeling a large complex system using abstractions such as diagrams. However, abstractions cannot, and presumably should not remove all complexity from the domain (Villberg, 2007). This result in a state where some complexities connected to the domain are reduced, but other aspects of the domain may increase the complexity of the interface. To cover as many scenarios as possible a broad range of functionalities are included in the system, and by extension the interface, thus increasing the complexity and resulting in a situation where only highly trained experts are able to effectively use the system (Arning et al., 2016).

Weyers, Burkolter, Luther, and Kluge (2012) explored individualization of UIs for handling complex systems. With their method, the physical representation of the interface and the behavior (interaction logic) was separated, rendering it possible to build custom interfaces and behavior that could be adapted to the tasks and knowledge. The study operates with the assumption that reducing user errors is the key to increase effectiveness and efficiency in interaction with the interface. With results indicating, that individualization would reduce the amount of general errors performed, with an increase in repetition errors. The authors considered individualization as a valuable method to support novices, as a form of training wheels, where the knowledge of the user is matched with the interface.

Furthermore, a lot of effort has been made to redesign and optimize GUI usability, along with development of standards, guidelines and principles (discussed in 5.1 Usability). However, a sizable amount of this effort, appears to be, focused on web-interfaces and interfaces for a mainstream audience (non-technical users), resulting in existing UI principles often being too generic for complex CAM systems (Arning et al., 2016), and provides little specific useful assistance with domain-specific factors.

Another proposition for usability is automation, such as automating the extraction of data from the technical applications. Carducci, Del Monaco, Giacchetta, Leporini, and Marchetti (2015) investigated automation to extract data from OLGA. The authors did not discuss GUI specifics, but investigated automation, with the aim to improve the process of extracting results from OLGA simulations. The current process of extracting results from OLGA requires that users to create trend and profile plots in OLGA in a process consisting of the following steps:

1. Visualize the variables of interest in a trend/profile plot
2. Use value tracking functionality to present data from selected points (hover and click operation)

3. Elaborate the information if needed
4. Process data

This is a process that requires a lot of time and effort while being slow and repetitive (Carducci et al., 2015), including a dependency on the hand-eye coordination of users, as well as precision and attention when manually locating the desired values, and when clicking on the points (in the plot) to define which data is needed. Because of the process, users must spend a lot of time to select what data to extract, requiring cognitive attention and an overview of where the desired data is in the plot, and risk precision errors, *i.e.* a slight movement of the pointer when clicking may result in the wrong values being presented. After the data has been extracted it is also likely to be manually arranged for specific purposes (*e.g.* creating a plot or for use in a report). With an automation tool, the accuracy and reliance on data extraction can be improved, as well as efficiency improvements, where the data can be automatically can be arranged in the desired format. The automation tool developed by Carducci et al. (2015) was shown to improve the data extraction time (speed up), where the manual extraction method took ten to fifteen minutes, the automation took one minute.

Literature focused on developing interfaces for other technical systems exists. Lado et al. (2006) developed an alternative interface, called R-interface, for the MATLAB environment with the aim of enhancing particular functionality for educational purposes. The authors illustrate a system based on integrating and reusing MATLAB, to develop a user-friendly GUI for use in classrooms. The authors targeted inexperienced users, and the GUI was developed to compromise between educational purposes and respecting MATLAB concepts and operation. The authors draw attention to the similarity to other widely used software, such as Excel, ensuring that the users are familiar with screen elements such as menu bars, icons, and popup help. Furthermore, a history list is included in the GUI, with all previously executed commands, the history enable easy to reversal of the system state to a previous one; the history enables re-executing and copying commands. History lists are also used in other widely used software such as Microsoft Word and Adobe Photoshop, and beyond enabling easily reversing multiple actions, without repeatedly clicking the undo button blindly. Furthermore, cognitive strain on the user is reduced, as there is less need to remember what actions has been performed. R-Interface includes a panel with a variable list, presenting all the variables with name, size and dimensions. To facilitate easy identification R-Interface use different icons for the various variable types, making it possible to identify types visually without interpreting the textual information. An interesting feature of R-Interface is the integrated COM interface, making it possible to integrate external wizards or assistants. Further possibilities with the COM interface, is interfacing other software environments with R-Interface, such as a wizard making it possible to create graphs in Excel with MATLAB data.

Other fields subjected to studies includes daylight simulation for building designs (Wu & Ng, 2001), and tools for DC-DC converter circuits and active power factor correction (Kayisli, Tuncer, & Poyraz, 2013). The study performed by Wu and Ng (2001) evaluated two different software systems in terms of accuracy of the simulation results and the performance of the software from the designers view. The accuracy of the simulation results were tested by collecting real data, from residential buildings in Hong Kong, and comparing the data to the simulation results. The performance of the software was evaluated by conducting a usability study, with two students with experience with AutoCAD. The software systems evaluated were “Desktop Radiance”, which was integrated with AutoCAD, and “Lightscape”. Due to Desktop Radiance being integrated with AutoCAD the students had the impression that it would be easier to learn, as almost every aspect of the software seemed familiar to them. The usability study revealed that the Lightscape software was preferred, after having mastered the system, favoring the flexible UI, and the system being intuitive, easy to learn and having a better UI. In comparison the then current version of Desktop Radiance was bug ridden, causing the students to lose interest quickly. Furthermore, the Desktop Radiance system’s interface was described as looking like scientific software. Of interest is the reasoning given by the students, where Lightscape would be preferred in initial design stage, due to rendered results being more useful in making design presentations. However, the students would not mind using Desktop Radiance in the final stage if it provided more information that is scientific. The results may indicate that users in technical domains may prefer the system providing the most value for the given situation.

Kayisli et al. (2013) developed a tool for educational purposes, where the aim was to assist in the education of concepts in DC-DC converter circuits and “active power factor correction applications”. The authors note that a benefit of GUIs was the ability to post-process the simulation results and provide instant feedback to the user, a feature important when conducting parametric studies. The developed interface supported multiple relevant concepts. As the aim was to support education in concepts, tools for modeling circuits were not included, instead the interface provided screen elements enabling the user to manipulate converter and controller values and receive visualizations of how the manipulations affected the current/signal.

7 OLGA

The name OLGA refers to a simulator engine (OLGA) and a GUI (OLGA GUI), in this report the name OLGA will be used to refer to the GUI, the simulator engine will be referred to as the simulator. The rationale is that this report focuses on the redesign of the UI.

7.1 Simulator

The OLGA simulator is a dynamic multiphase simulator capable of predicting steady-state and dynamic multiphase flow across a range of production conditions. The simulator has been continuously developed and improved with ongoing field and lab research for over 30 years, which has helped refine the simulator to provide accurate predictions of steady-state and dynamic multiphase flow (Schlumberger Software, 2015).

The simulator is used in flow assurance to simulate the simultaneous transportation of oil, gas and water (multiphase fluids) through the same pipeline. Flow assurance focus on the transport of multiphase fluids from reservoir to refinery, where the optimal is to ensure continuous and economical flow of fluids (Tine Bauck Irmann-Jacobsen & Hægland, 2014). The simulator can be used in different stages of the life cycle of oil/gas transportation, such as in the development of production systems, planning and development of operational procedures. The simulator can also be integrated into online operations, where it can be used to monitor the system, and offer information relevant to a range of scenarios with real-time look-ahead and what-if modes.

When planning new production systems, the simulator can be used to predict how fluids will behave under various conditions such as temperature, geometry, pressure, and much more. With this functionality, it is possible to test alternatives and decide how to build the new system. For example, while planning a production system it is possible to test multiple different inner pipeline diameters, and select the ones with the optimal flow rate. It is also possible to predict how different materials and conditions affect the flow. When procedures, such as a system shutdown, starting a system, or pigging operations are planned, OLGA can be used to simulate the order in which the components in the system is shutdown, and later in which order the system should be started up, in these scenarios the inside and outside conditions in the system can cause damage to equipment. The simulator can further be used to detect conditions where slugs can form, as well as wax depositions (Aiyejina, Chakrabarti, Pilgrim, & Sastry, 2011). Slugs are challenging flow patterns occurring in pipelines transporting multiphase fluids, and can cause irregular flow, periods without production, system shutdown and damage to equipment and pipes (Airam Sausen, Paulo Sausen, & Mauricio de Campos, 2012; Statoil, 2009).

Simulations are performed on cases which include a model of the production system and definitions of material used, inside and outside conditions (for example temperature and pressure), as well as geometry. The definitions are stored in text format (ASCII) in a file that is used as input for the simulator.

The GUI assists in the creation of cases by enabling the users to create a visual model of the production system in a diagram, define materials and conditions, as well as plot the simulation results to profile or trend plots. Before the GUI was developed, the users had to define the cases manually in text files.

7.2 Background

The development of OLGA started in 1980 by IFE, Statoil financed the development in the early stages, and the early versions of OLGA were used by Statoil to analyze practical problems (IFE, n.d). From 1983 to 1992, IFE and SINTEF started collaborating, dividing the responsibilities so that IFE developed the code, while SINTEF created empirical data through laboratory experiments, to be used by OLGA. In 1993, Statoil entered an agreement with IFE and SINTEF. The agreement was to focus on multiphase research and development (R&D), the agreement lasted five years, Norsk Hydro and Saga Petroleum joined the agreement for the last two years. The result from the agreement was handed over to the OVIP project, which resulted in the OLGA 2000 version. In 2003, IFE and SPT started the HORIZON project, with the aim of replacing empirical correlations with models based on physics to improve reliability when scaling up from laboratory to field scale (IFE, n.d).

Commercialization of OLGA began in 1993, when commercialization rights were given to Scandpower AS (later SPT Group AS) and in 2012 SPT Group and OLGA was acquired by Schlumberger (IFE, n.d; Schlumberger, 2012).

7.3 Brief introduction to working with OLGA

When working with OLGA the user create and defines a *case,* that includes a model of the system, properties of components, inside and outside conditions (of the pipeline) and output variables to perform simulation on, *i.e.* define which results are relevant. It is also possible to create a *project* containing one or more *cases*, a project can be useful when working with large systems, which can be split up into multiple cases. A project is a collection of cases that indicates which cases to load when the project is opened. When a case is created, OLGA will create a temporary project that later can be saved or discarded, without discarding the case. There are two alternative methods available when creating a case: *creating an empty case* and *creating a case from a sample*. An empty case provides a blank canvas with no predefined properties. However, some basic output variables are defined. Sample cases provide a complete model, including predefined properties and relevant output variables, making sample cases ready for simulation immediately.

It is not required to create a project, but can be useful on larger systems where the model can be split up into multiple cases. If a project is not created (or previous project active), OLGA will default to the case name, of the first case opened in OLGA, as a temporary project name. Furthermore, when saving the project, it will be saved in the same directory as the first case.

A case contains the model and definitions of materials, walls, initial and boundary conditions, simulation settings and definition of output variables and much more. Each component in the model includes a set of properties that can be defined, and flow-paths (pipelines) include additional sections for output variables, making it possible to obtain the behavior locally.

OLGA includes three different modes for performing simulations, batch, interactive and step. The batch simulation launches a command line where the cases are simulated independently from the OLGA GUI. Interactive are performed in the GUI and step simulations is an extra functionality using the interactive simulation. An interactive simulation performs the complete simulation in one go and can be paused, restarted and stopped. Step simulations perform simulations in time steps that can be defined in the case properties, and in a drop down dialog on the step simulation button on the toolbar. Simulation results, from all modes, are stored in separate files, named after the case file they belong to the file name is the only association the result files have to a case.

Plotting functionality is integrated in OLGA, using a third party library. With the functionality, it is possible to plot the simulation results in trend plots that visualize the behavior of time varying variables over time at a given location, and profile plots, visualizing the behavior along a distance. The plotting functionality use line charts to visualize the trend and profile variables.

Furthermore, OLGA provides two modules for plotting, which are referred to as *interactive* and *static* plots in this report. The latter is not used by OLGA, and is only used to separate the modules in this report, the naming is not intended to imply any characteristics of the plots, but are based on where the modules obtains the data that is plotted. An interactive plot retrieves and plots the data straight from the simulator, which requires an interactive simulation. A static plot retrieves and plots the data from the result files, and can include other files not belonging to the active case. Static plots can plot the data from any of the simulation modes, provided the simulation has completed and the results are stored in files. The behavior differs between the modules when closing a plot tab, case, project or OLGA. Interactive plots store the layout and variable selections with the case, but require an interactive simulation to plot the data again. Static plots are not stored anywhere with the result that all plots are lost when the case is closed.

7.4 File Handling

While working with a projects and cases, OLGA creates various files, serving different purposes. These files are used to store case definitions, models, projects and simulation results as well as simulation output. Some of the file types that OLGA use are introduced in Table 7-1 below, and may vary depending on the version of OLGA.

Type	Extension	Description
Case	OPI	[GUI] File containing the model and definitions
Project	OPP	[GUI] File containing project details, mainly paths to Case files
Input	GENKEY	[SIM] File with model and definitions
PVT table	TAB	ASCII Table of fluid properties, required for simulation
Batch	BAT	[SIM] Indicates which Input files to include in a batch simulation
Trend	TPL	ASCII file storing simulation results for trend variables
Profile	TPP	ASCII file storing simulation results for profile variables
Output	OUT	Report generated by the simulator

Table 7-1: Files used by OLGA

Fluid property tables are stored in .tab files, and are required to perform simulations. OLGA includes a bundle of internal and external tools covering different purposes; one of the external tools is used to assist in creating the PVT (Pressure-Volume-Temperature) table, creating a Case from a sample includes a predefined PVT table. It is also possible to create the file manually, but due to the complexity and ASCII format, this requires a lot of effort and may not be feasible.

The model and belonging properties created with OLGA are stored in the XML-format in case files. The files are used to describe the structure of the model and are used by OLGA to draw the diagram.

The simulator uses the .genkey files and not the .opi files when performing simulation. The case and input files (.opi, .genkey) are automatically created by OLGA when creating a new case. Case files do not know if they belong to a project, which may result in a case being shared between multiple projects.

The project files (.opp) link the .opi files belonging to the project. OLGA does not know if a case belongs to a project unless the project file was opened. The project file use relative paths based on the project file’s location, this means that when moving project files then all belonging case files must be moved and directory structure must be maintained.

Simulation results are stored in the appropriate trend or profile files, as ASCII, which requires more disk space, and memory to load.

7.5 OLGA's Requirements

The users of OLGA are technical domain users, working with flow assurance or other relevant areas in the oil and gas industry. In many situations, the purpose of using OLGA is to discover or anticipate how various situations and conditions, affect the transportation of multiphase fluids. Essentially the purpose of use is to discover what is *not yet known*; as such, the users may not know what they are looking for or what they need to understand the results. However, OLGA requires that the users know in advance what they are looking for and need before performing a simulation.

Aside from requiring domain knowledge, OLGA requires that users define the desired output variables, to obtain the data necessary to understand the behavior in the production system. For inexperienced users this may result in a lot of trial and error, and may be problematic for more experienced users. Users may spend a lot of time trying to anticipate the needed variables necessary to understand and explain the behavior. This can be costly in terms of time, frustration and accuracy, more complex cases require more time to complete a simulation, and may require restarting the simulation if a variable was not specified. Restarting a simulation may not be desired or possible, due to time constraints.

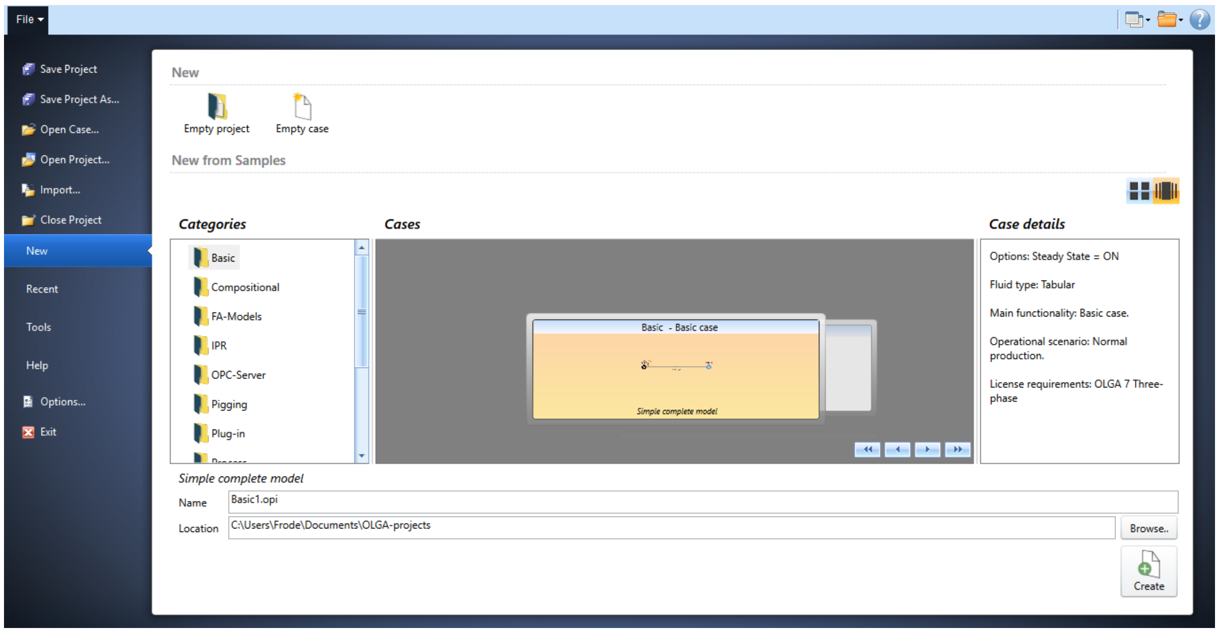
8 Graphical User Interface

In this chapter, the GUI is described, with a focus on what the interface provides, as well as behaviors and some comparison with platform conventions. Additionally, some potential issues, such as behavior, are highlighted in the context. The chapter is split into separate sections for file menu, main interface and plot view.

When OLGA is started, the user is presented with the file menu with the new-page active, which functions as a start screen. The main interaction with OLGA occurs in the main interface where the "diagram view" and plot view is located.

8.1 File Menu

By default, the file menu presents a full screen menu (Figure 8-1). The layout of the file menu consists of a button pane on the left side with actions and tasks, and a pane for *pages* that uses the remaining space. A *page* contains actions related to the currently active tasks. An overview of the entries in the button pane and type of dialog displayed when activated is provided in Table 8-1.



Options: Steady State = ON

Fluid type: Tabular

Main functionality: Basic case.

Operational scenario: Normal production.

License requirements: OLGA 7 Three-phase

Simple complete model

Name: Basic1.opi

Location: C:\Users\Frode\Documents\OLGA-projects

Figure 8-1: OLGA's file menu and start screen

Action	Description	Dialog
Save Project	Saves the current project and all open cases	None
Save Project As	Saves the project and all open cases with a new name	File-dialog: save
Open Case	Open a case file (.opi)	File-dialog: open
Open Project	Open a project with all belonging case files	File-dialog: open
Import	Import a case from input file (.geninp)	File-dialog: open
Close Project	Close current project and all open cases	Prompt
New	Create a new case or project	Page
Recent	Lists recently used projects and cases	Page
Tools	Lists internal and external tools	Page
Help	Displays help options and information about OLGA	Page
Options	OLGA Options	Option window
Exit	Exit OLGA	Prompt
Dialog types	File-dialog: standard Windows file choose dialog Page: options/actions presented in the OLGA UI Option window: options displayed in a separate window Prompt: prompts user if unsaved changes exists	

Table 8-1: List of action entries in the file menu

8.1.1 Actions

When projects are created a file-dialog for saving is displayed, making it possible to specify the project name and storage location. Cases created after creating or loading a project will be stored in the project directory. If there is no active project and a case is created, OLGA creates a temporary project and new cases will be stored in the same directory as the first case, unless otherwise specified.

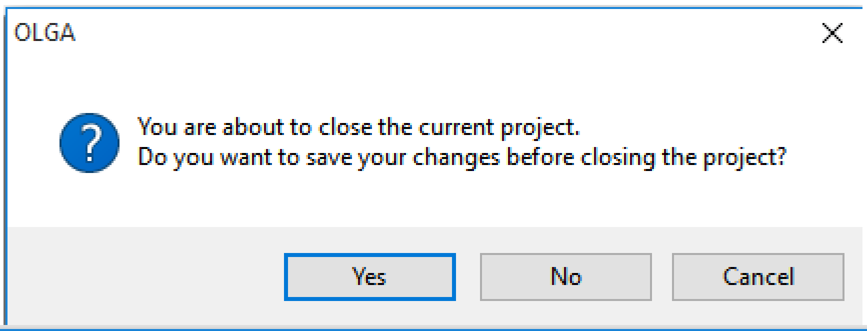
There are two options available when creating cases *empty case* and *case from sample*. When an empty case is created, the user has no control over the case name or location. The file name used is "case", and an incrementing number is appended, if the file name already exists in the directory). Changing file name can be achieved by manually renaming the file outside of OLGA or using save as (on the toolbar), which results in a copy of the case. Either the location is the location of current active project or, if no projects are active, a directory specified in the OLGA options. When a sample case is created, the controls are provided in the new-page to specify the case name and location. The available samples are organized in categories.

When **save project** is used, there is never any form of dialog, which is expected when a project was explicitly created. However, when saving a temporary project the user have no control over the project name, which may not be descriptive, or the location. Furthermore, the action is can only be used when there are changes to the project, such as adding new cases, and is disabled when changes do not affect the project file, changes to cases belonging to the project do not enable the button. The save project action is also assigned a keyboard shortcut (ctrl + shift + s) that is not advertised in the interface but mentioned in the documentation. However, the documentation incorrectly informs that the keyboard shortcut is assigned to *save project as* but observing the behavior shows that the action *save project* is used (no dialog appears when using the shortcut). Furthermore, users familiar with the Windows platform may be used to the shortcut as the *save as* alternative to ctrl + s (in OLGA this is save case), this is however not a requirement.

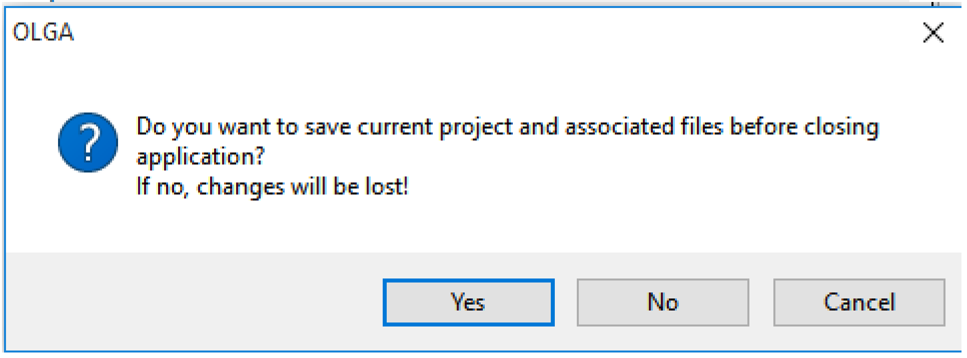
When **save project as** is used, a file-dialog is shown to create a copy of the project file with a different location and name. Copies of case files are not created, resulting in multiple projects sharing the same case files, unless the initial project was a temporary project. The action is not suited for creating *complete copies* of projects, but can be used to achieve more control when saving temporary projects.

Open case and **open project** are similar actions with slightly different behavior, both actions close the file menu and display a file-dialog above the area of the diagram view. Open case is used to load a single case into OLGA, while open project loads all cases associated with the specified project. The **import** action can be used to import input files (.genkey) which is used to create a new OLGA case.

When **close project** is used the user is prompted to save changes, if there are unsaved changes to the *project*, such as adding or removing cases to/from the project. The prompt is not displayed for unsaved changes affecting only cases (such as modifying the model). The prompt asks if the user wants to save the changes before closing the project, the available options are *yes*, *no* and *cancel*. When closing OLGA a similar prompt is displayed, with the same behavior, where modifications to the project trigger the prompt and changes to cases are ignored. Furthermore, the prompt is also displayed even when a project has not been opened or explicitly created.


OLGA

You are about to close the current project.
Do you want to save your changes before closing the project?


OLGA

Do you want to save current project and associated files before closing application?
If no, changes will be lost!

Figure 8-2: Prompts displayed when closing a project (top) and closing OLGA (lower)

8.1.2 Pages

The file menu includes pages for the following entries in the button pane:

- New (new-page)
- Recent (recent-page)
- Tools (tools-page)
- Help (help-page)

New-page (Figure 8-1) – The layout of the page is organized in two sections separated vertically. The section labeled “New” contains two buttons used to create empty projects and empty cases. These buttons illustrate a particular problem with a majority of the buttons located in the pages, though it is more frequently experienced with the buttons located on the new-page. The problem is that the pointer must be within a specific area to perform the desired action when clicked. The buttons have two hover states (Figure 8-3), when the pointer hovers over the general area of the button, the background will change to blue. However, clicking in that area will not initiate any action, when moving the pointer a little more to the center of the button an additional yellow background color is added and clicking will initiate the action.

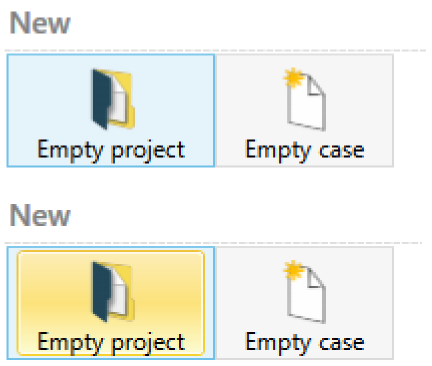


Figure 8-3: hover states of buttons for empty project and empty case

The lower section labeled "New from Samples" consists of a sample selector, text input for case name, input control and button to a directory picker to specify location and a button for creating the case. When manually entering a path in the location input, the value is not used when pressing the return key on the keyboard, unless the focus has left the input field. The sample selector has two modes, the default preview and icon view (Figure 8-4) icon view reduces the visible information. The samples are organized in categories, available from a list to the left of the preview, and a case description located on the right of the preview. Between the three elements are vertical separators that can be resized without any constraint.

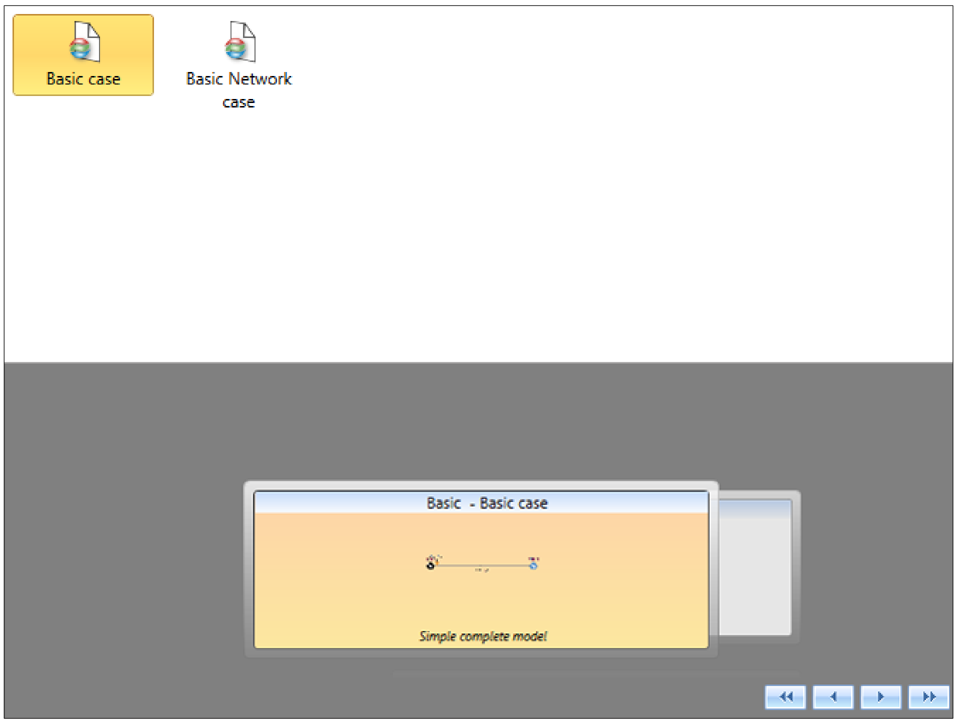
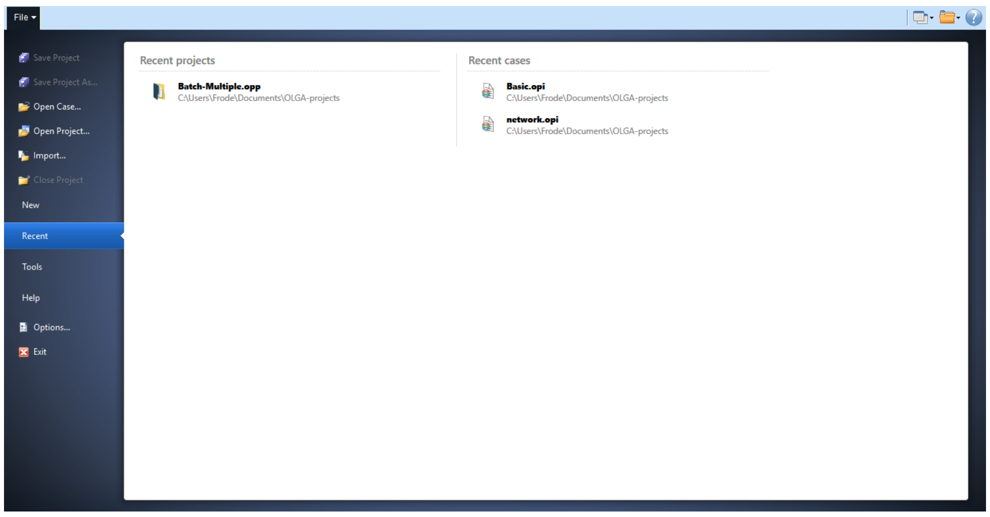


Figure 8-4: cropped comparison of sample selector’s icon and preview modes

Recent-page (Figure 8-5) – The content on this page is organized in two horizontally separated panes, with a separator in between. The left pane lists the recently used projects and the right pane lists the recently used cases. The separator can be resized with virtually no restraint, resizing to the left is only stopped by the button pane, while resizing to the right can continue beyond the edge of the window with no method to revert the resizing except restarting the application. An intermittent issue with the recent-page is that it sometimes appears to be disabled (entries grayed out) even though it is not.



Recent projects

Batch-Multiple.opp
C:\Users\Frode\Documents\OLGA-projects

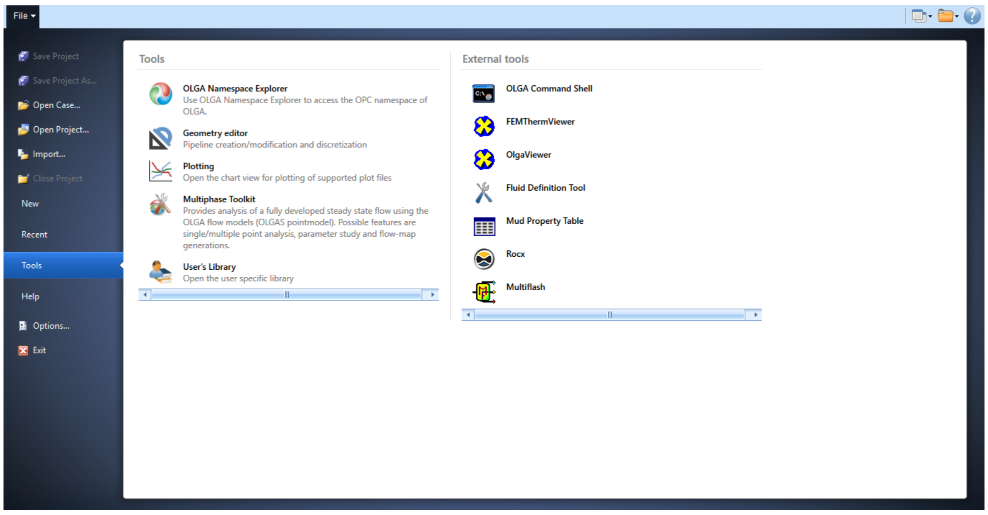
Recent cases

Basic.opi
C:\Users\Frode\Documents\OLGA-projects

network.opi
C:\Users\Frode\Documents\OLGA-projects

Figure 8-5: recent-page listing recently opened projects and cases in separate panes

Tools-page (Figure 8-6) – The layout organizes the tools in two panes with the categories *tools* and *external tools*. Some external tools may require additional licenses that is not indicated, and only discovered when the tool fails to start. Horizontal scroll bars are, by default, visible adding unnecessary visual noise. The separator between the two panes can be resized, with constraints preventing unlimited resizing.



Tools

OLGA Namespace Explorer
Use OLGA Namespace Explorer to access the OPC namespace of OLGA.

Geometry editor
Pipeline creation/modification and discretization

Plotting
Open the chart view for plotting of supported plot files

Multiphase Toolkit
Provides analysis of a fully developed steady state flow using the OLGA flow models (OLGAS pointmodel). Possible features are single/multiple point analysis, parameter study and flow-map generations.

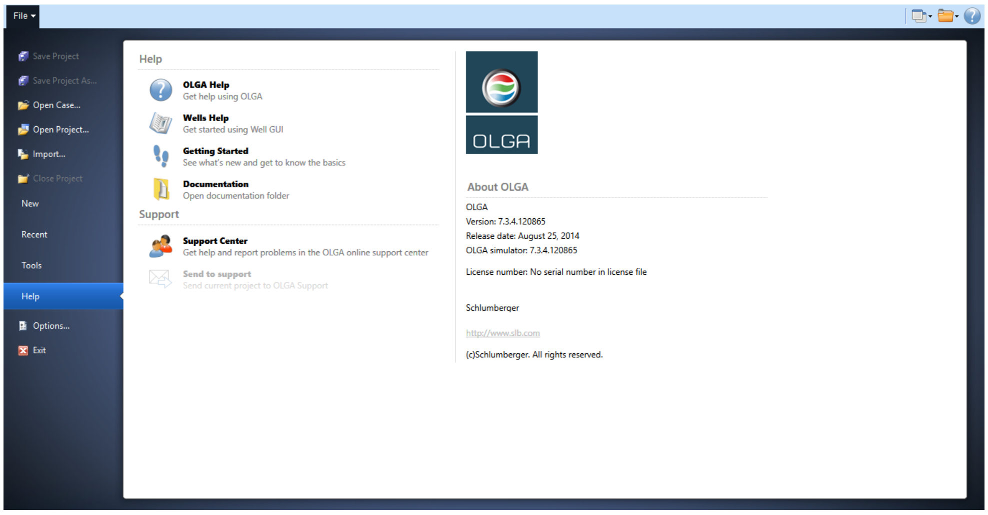
User's Library
Open the user specific library

External tools

OLGA Command Shell
FEMThermViewer
OlgaViewer
Fluid Definition Tool
Mud Property Table
Rocx
Multiflash

Figure 8-6: tools-page listing internal and external tools in separate panes

Help-page (Figure 8-7) – The layout consists of two horizontally separated panes, the left pane consists *help* and *support* categories, while the pane on the right by default consists of *about OLGA* information, which is replaced when getting started is selected (Figure 8-8). The help category lists the topics *OLGA Help* and *Wells Help*, *Getting Started* and *Documentation*. The support category lists *Support Center* and *Send to Support*.



Help

OLGA Help
Get help using OLGA

Wells Help
Get started using Well GUI

Getting Started
See what's new and get to know the basics

Documentation
Open documentation folder

Support

Support Center
Get help and report problems in the OLGA online support center

Send to support
Send current project to OLGA Support

About OLGA

OLGA
Version: 7.3.4.120865
Release date: August 25, 2014
OLGA simulator: 7.3.4.120865

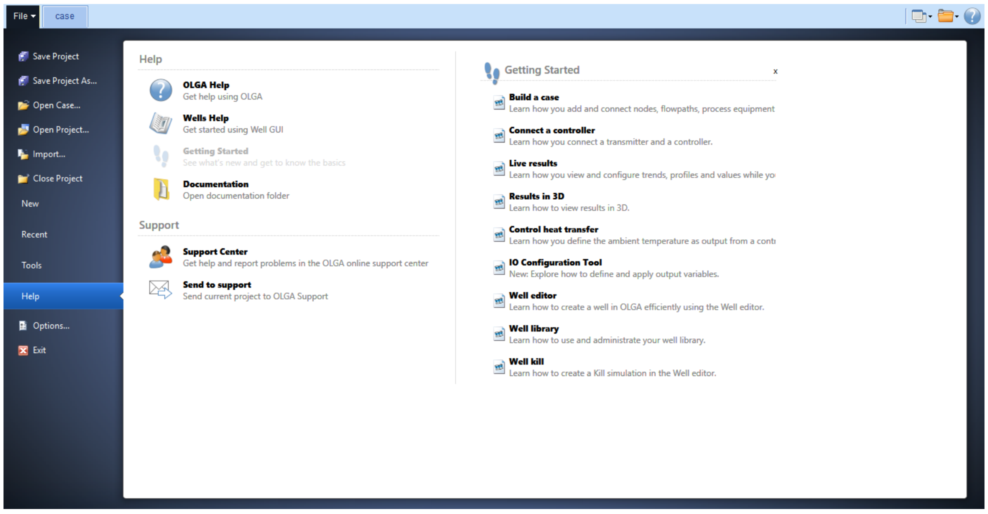
License number: No serial number in license file

Schlumberger

http://www.slb.com

(c)Schlumberger. All rights reserved.

Figure 8-7: help-page's main screen with help topics and information about OLGA



Help

OLGA Help
Get help using OLGA

Wells Help
Get started using Well GUI

Getting Started
See what's new and get to know the basics

Documentation
Open documentation folder

Support

Support Center
Get help and report problems in the OLGA online support center

Send to support
Send current project to OLGA Support

Getting Started x

Build a case
Learn how you add and connect nodes, flowpaths, process equipment

Connect a controller
Learn how you connect a transmitter and a controller.

Live results
Learn how you view and configure trends, profiles and values while yo

Results in 3D
Learn how to view results in 3D.

Control heat transfer
Learn how you define the ambient temperature as output from a conti

IO Configuration Tool
New: Explore how to define and apply output variables.

Well editor
Learn how to create a well in OLGA efficiently using the Well editor.

Well library
Learn how to use and administrate your well library.

Well kill
Learn how to create a Kill simulation in the Well editor.

Figure 8-8: help-page getting started screen with a list of instructional videos

When the content on the right side is changed it also produces a noticeable vertical shift on the left pane (Figure 8-9), which when occurring may appear to be removing an entry from the list.

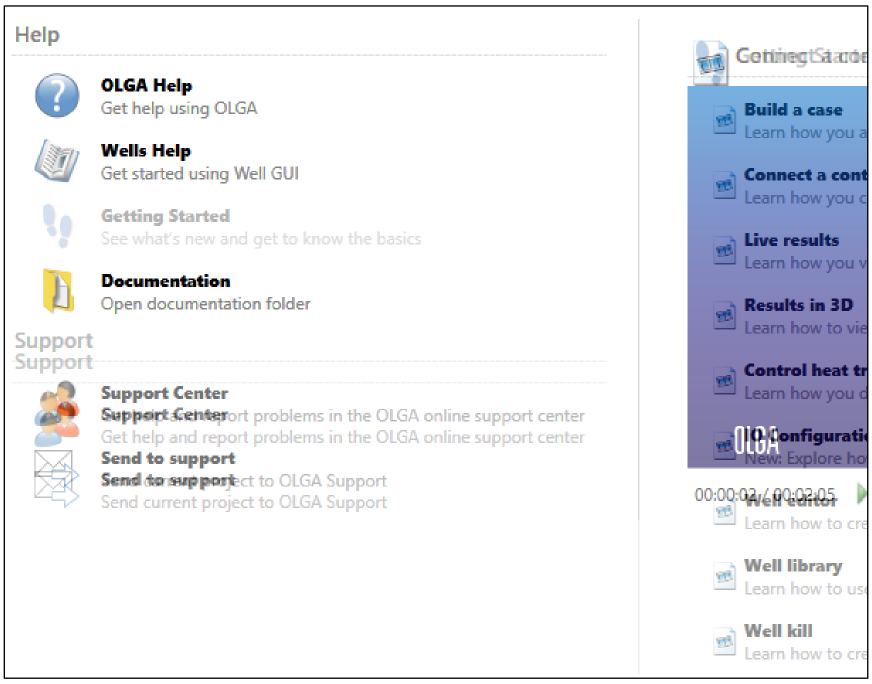


Figure 8-9: overlay of pages illustrating a vertical shift of help topics

The entries for *OLGA Help* and *Wells Help* opens separate help windows, the *Documentation* entry launches a file browser at the location of the printable versions of the manuals. The entry for support center launches a web-browser navigated to Schlumberger's website, and *Send to support* launches an email client (if installed). Furthermore, as can be seen in Figure 8-7 and Figure 8-8 the text color used when the *send to support* button is disabled have too low contrast with the background, likewise, the entry for getting started has too low contrast in the text color used when the getting started entry is *focused* (Figure 8-8, Figure 8-9).

8.2 Main Interface

When a case is open, the user is presented with the main interface (Figure 8-10), which by default consists of a pane for case-tabs, the toolbar, a pane model-tabs, the diagram view, components view, model browser and a collapsed output tab, and a status bar.

The pane with case-tabs consists of the tabs for the opened cases, a file menu button and dropdown menus for *view*, *project* and *help*. The toolbar consist of icon buttons for functionality that is primarily intended for the diagram view. Tabs that are used for plots, and other tools that may be inserted as a tab, are inserted in the pane with the model-tab. The diagram view consists of the canvas where the model is created, the component view and model browser floats above the diagram view. The component view provides the components that can be used to create the model, while the model browser provides a tree view and property editor used to define the model and case. Below the diagram view is the output tab that is by default collapsed. The status bar contains an indicator for the simulation state and functionality to zoom and pan the diagram view.

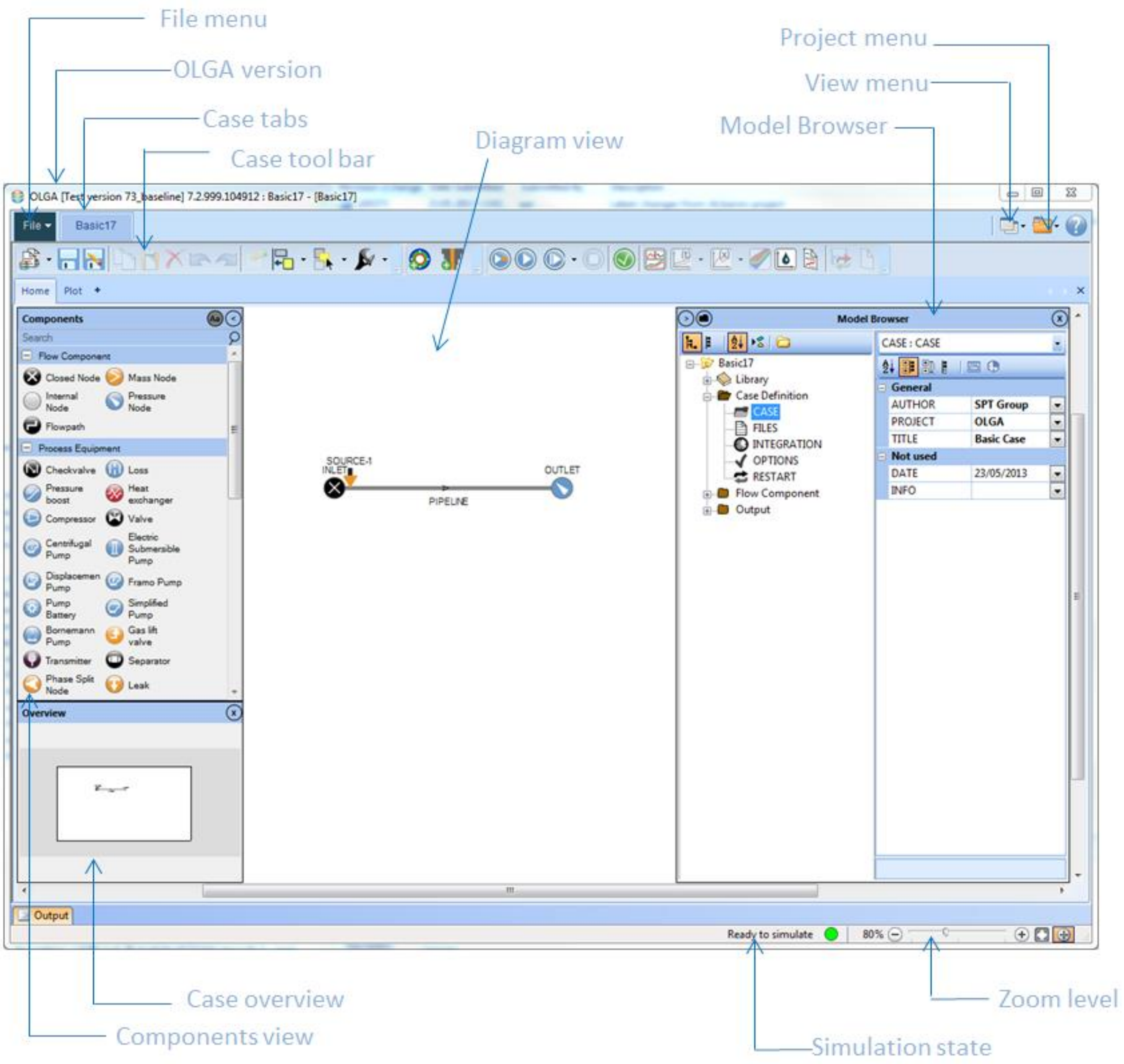


Figure 8-10: main interface with names of elements

Figure 8-10 includes a floating window for Case overview that provides an overview of the complete model with a miniature version of the model, which can be useful when working on large models. The Case overview window is not visible by default and must be manually activated in the *View menu*.

Figure 8-10 is taken from the OLGA user manual (*User Manual OLGA 7.3*, n.d).

8.2.1 Case Tab Bar

The case tab bar includes a button for the file menu, case tabs and dropdown buttons for *view menu* and *project menu*, and a *help button* to launch the OLGA manual.

The OLGA **help button** opens the same manual that the “OLGA Help” entry in the Help-page opens. However the *window* behavior is not consistent, “OLGA Help” opens an independent window where it is possible to use `alt + tab` to shift focus between the manual and OLGA. The toolbar button opens a window that depends on the main window, meaning that to shift focus between the manual and OLGA the manual must be minimized/restored manually.

The **project menu** includes the following:

- Run project – performs an interactive simulation on all cases in the project
- Run project batch – performs a batch simulation on all cases in the project
- Add existing item – can be used to add any type of file to project
- Project dependencies – opens a dialog displaying the project dependencies
- Close project

The **view menu** includes the following toggle entries:

- Model view – tree view of model located in model browser
- Properties – property editor located in model browser
- Output – output tab located on the lower tab-bar
- Components – floating window with components
- File view – switches the model browser model view with file view
- Connections – adds a new tab on the lower tab-bar with the output tab
- Overview – adds a floating window with overview

As could be expected of the view menu it contains entries to add/remove elements to the interface, by default the model view, properties, output and components are activated. The view menu does however have some unexpected behavior with model view and file view, if file view is activated it replaces model view and vice versa. It is also possible to deselect everything except properties and model view (or file view if activated), to hide the properties the model browser window must be "closed".

The **case tabs** are simply tabs for the cases, there is no close button located on the tab. The context menu on right click does not have anything to do with the tab. The context menu can be used to add a main menu above, or plug-in instances below the tab-bar.

The added main menu is not a main menu but a menu serving the same purpose as the case tabs. The plug-in instances are adequately advertised on the toolbar.

To close a case tab the user has three options, close the OLGA application, close project, or locate the corresponding functionality on the toolbar.

The toolbar consists of icon buttons to complete certain tasks; the buttons have tooltips indicating what functionality it provides.

From the left we have, a button for duplicating a case with a dropdown menu including an entry for removing the case.

Duplicate case enables the user to create a new case based on an existing one and is similar to a save as option, except that duplicate case also opens a new case tab for the duplicate.

Remove case is the corresponding action to close a case tab, and enables the removal of a case from a project this action provides the following options:

- Remove case from project
- Remove case from project and delete case file
- Remove case from project and delete all associated files
- Cancel

From the listed options, there is no alternative for closing the case tab without removing it from the project, which may be desired when working on large projects with many tabs.

The **save case** button is used to save changes on a single case. The button tooltip advertises the keyboard shortcut `ctrl + s` a standard shortcut for this action.

The **save case as** button is used to create a copy of the specified case, and after use will replace the original case with the new copy.

Some other standard actions available on the toolbar are **copy**, **paste**, **delete**, **undo**, and **redo** these actions are only possible to use when working on a model.

The **properties** button's functionality is dependent on the currently selected component in the diagram, for some components it has no apparent function, for a flow-path it launches the geometry editor, for a centrifugal pump it launches another window. The other windows the properties button launches are not visibly advertised to the user anywhere in the GUI.

The **layout** button provides functionality to toggle if a grid should be displayed on the diagram view, toggle snap-to-grid functionality, if flow-paths should be straight lines or use right angles. It also includes functionality to arrange the model horizontally or vertically as well as fit-to-page that zooms and pans the diagram view so the whole model is visible.

The **current item** button includes entries for local instances, global instances, unlock, distribute inline equipment and duplicate all flow-paths.

The **tools** button includes entries for network connections, sub-model connections, adding components to a user library, import from user library and show the user library. As well as entries for parametric studies, copy model as image, launch RMO interface, IO configuration tool and fluid definition tool.

Two buttons to add **FEMTherm** and **OLGA Well**, which are tools used to assist in the creation and definition of models.

Three different buttons for starting different type of simulation, namely **batch**, **interactive** and **step**. Step simulation includes a dropdown menu where the step length can be defined. A **stop** button is also included which is disabled until an interactive or step simulation is performed. When an interactive simulation is running the interactive button also functions as a pause button.

A **verify** button is also included, which can be used to detect issues with the model definition. The verify function does not always work. In some situations, apparently only when missing required input, it will print the issues in the output tab along with a button that shift the focus to the location in the property editor. In other situations the simulator status will say “Not Runnable” and using verify will not produce any output.

The toolbar includes five buttons for different types of plots with tooltip labels: “Multiple plots”, “Trend plot”, “Profile plot”, “3D plot” and “Fluid plot”.

The button for “Multiple plots” adds a plot tab for an interactive plot to the case, by default the plot is empty with the title “Custom plot”.

The buttons “Trend plot” and “Profile plot” add tabs for static plots of trend and profile variables respectively. A fluid plot is similar to a profile plot and is used to visualize the selected fluid properties over a range of temperatures. The fluid plot can be used to create animated visualizations at different pressures.

The button, “3D view”, is used to add a three-dimensional representation of flow-paths to the case. The functionality can be used to render a visual representation of the flow-path and the behavior of fluids in the flow-path.

A **report** button is also included on the toolbar. The report functionality gathers the data from the case and generates an input report that is opened in a web-browser. The report does not include simulation results.

The two remaining toolbar buttons are **save restart** and **open output**. Save restart is used to save the paused state of an interactive simulation so that it can be started at the same position later. Open output opens the simulation report (.out) in the application defined to handle .out files in the OS settings.

8.2.2 Tab Bar

The model tab bar is a tab bar where the tab for the model is located, as well as tabs for the various available plots. Some tools such as the OLGA Well tool also add a tab to this tab bar.

Located beside the last tab, from the left, is an add button with a + symbol. The add button can be used to add interactive plot tabs to the tab bar.

Located at the right edge of the tab bar, are disabled arrow buttons which are enabled when the amount of tabs exceed the available width. To the right of the arrow buttons is a close button with an X symbol that can be used to close almost all tabs, except the model tab. The close button can also be used to close the OLGA Well tab. However, the closed tab will still be present when the case is reopened.

8.2.3 Diagram View

The diagram view is simply a canvas for the diagram that represents the production system model. The canvas support drag and drop from the components view. The diagram view also includes a context menu for clicked components, with entries varying depending on the type of component.

There is also a context menu for the diagram view itself, this context menu contains entries for arranging the diagram horizontally and vertically, fit-to-page, toggle grid visibility, network and sub-model connections, paste, undo, redo, copy as image and filter layout. All entries are available from the toolbar except filter layout, which can be used to filter what components are visible on the canvas.

The filter layout (Figure 8-11) functionality has a reverse function of what may be expected on pre-conceived behavior. The assumed behavior, based on the grouping of the *show all* entry, is that when entries are activated then the corresponding components are *shown*. However, the actual behavior is that when entries are activated then the corresponding components are *hidden*. Furthermore, the *show all* may be assumed to function as a toggle entry that can be selected, in reality it is only used to deselect all other filters.

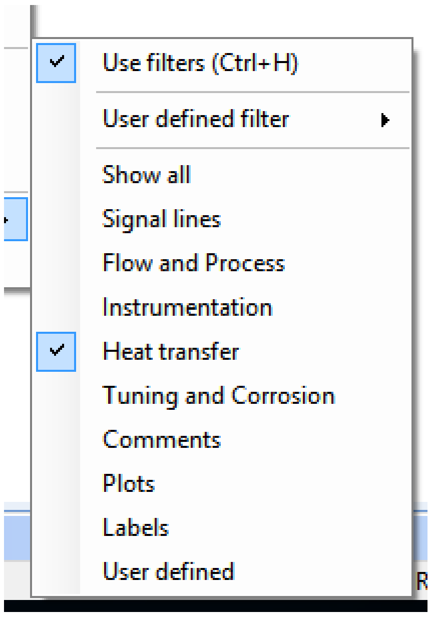


Figure 8-11: filters used to toggle the visibility of components

8.2.4 Components View

The components view is a floating window that can be minimized to a vertical tab on the left edge of the diagram view.

The floating window includes a button to toggle label visibility and a search bar to which performs a search on component names.

The components are organized according to the type of component such as flow components, process equipment and other categories. The categories can be collapsed and expanded to hide components.

The window can also be resized which re-order the components according to the available width. The window has a minimum width, and if the window width is resized down to a specific width the labels are automatically hidden, and the scrollbars are replaced with up and down buttons at the top and bottom of the window. To reach that state however, some fiddling with resizing is necessary.

The components are split into six different groups, with different purposes.

- Flow component
- Process Equipment
- FA-Models
- Boundary and initial conditions
- Controller
- Results and Comments

The **flow component** group covers nodes and flow-paths. **Process equipment** covers all equipment. **FA-Models** cover pigs, tuning and corrosion. **Boundary and Initial conditions** cover, according to the manual "*…only boundary conditions keywords and the comment field…*" (*User Manual OLGA 7.3*, n.d). The **controller** group covers all types of controllers that can be used in the model.

The **results and comments** group cover interactive plots, values and comments. The interactive plots can be added directly on the model (Figure 8-12), and have the same functionality as interactive plots added as tabs.

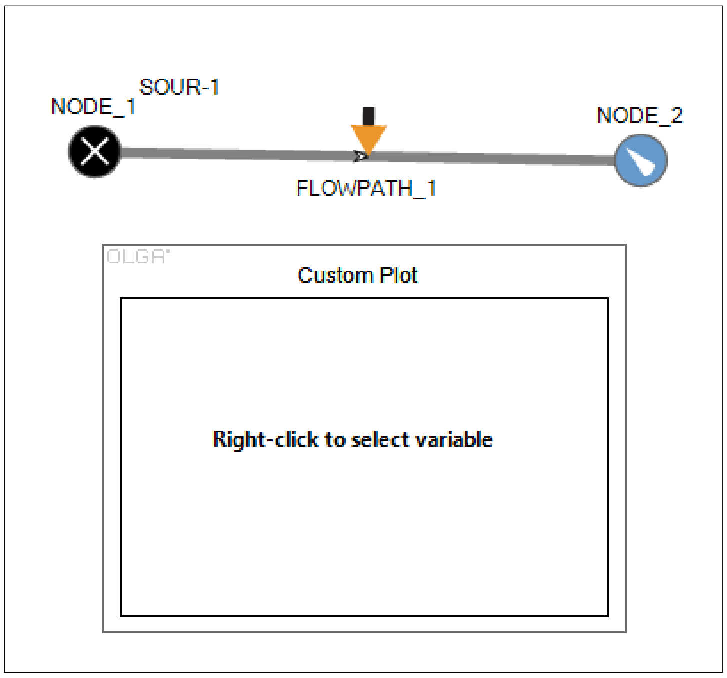
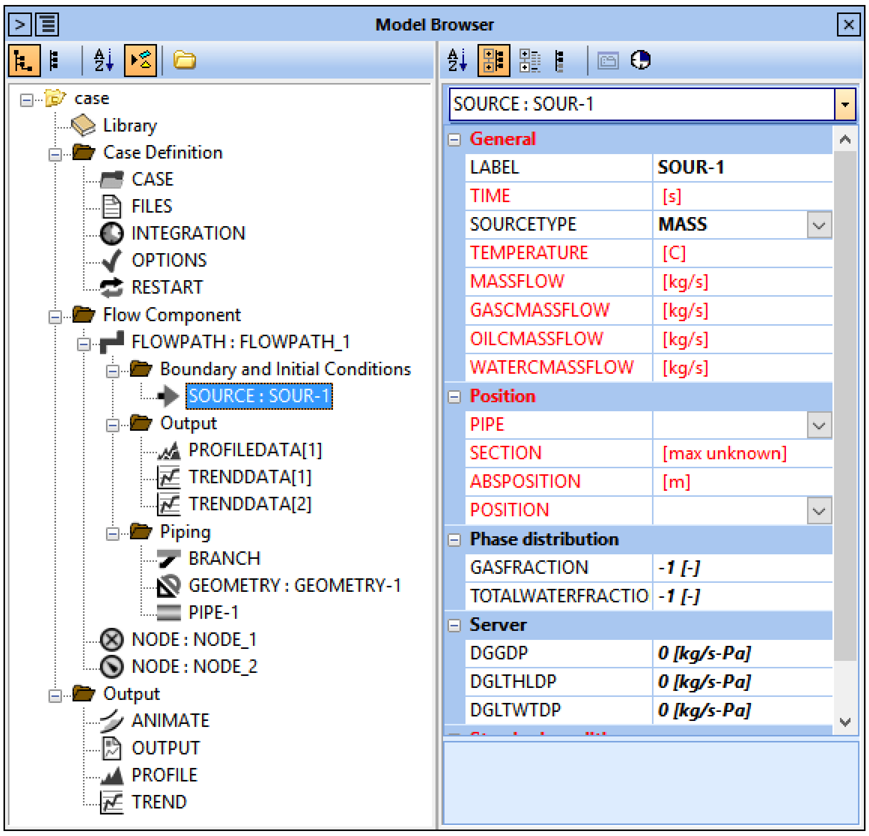


Figure 8-12: illustration of interactive plot added in the model

8.2.5 Model Browser

The model browser (Figure 8-13) is a floating window consisting of two panes: the **navigation pane** and the **property editor**.



Model Browser

```
case
  Library
  Case Definition
    CASE
    FILES
    INTEGRATION
    OPTIONS
    RESTART
  Flow Component
    FLOWPATH : FLOWPATH_1
      Boundary and Initial Conditions
        SOURCE : SOUR-1
      Output
        PROFILEDATA[1]
        TRENDDATA[1]
        TRENDDATA[2]
      Piping
        BRANCH
        GEOMETRY : GEOMETRY-1
        PIPE-1
    NODE : NODE_1
    NODE : NODE_2
  Output
    ANIMATE
    OUTPUT
    PROFILE
    TREND
```

SOURCE : SOUR-1

General	
LABEL	**SOUR-1**
TIME	[s]
SOURCETYPE	**MASS**
TEMPERATURE	[C]
MASSFLOW	[kg/s]
GASCMASSFLOW	[kg/s]
OILCMASSFLOW	[kg/s]
WATERCMASSFLOW	[kg/s]
Position	
PIPE	
SECTION	[max unknown]
ABSPOSITION	[m]
POSITION	
Phase distribution	
GASFRACTION	***-1 [-]***
TOTALWATERFRACTIO	***-1 [-]***
Server	
DGGDP	***0 [kg/s-Pa]***
DGLTHLDP	***0 [kg/s-Pa]***
DGLTWTDP	***0 [kg/s-Pa]***

Figure 8-13: screenshot of model browser with tree view expanded

In the title bar of the floating window there are three buttons, a button to collapse the navigation pane, a button to toggle between the model view and file view in the navigation pane, and a button to “close” (minimize) the model browser.

8.2.5.1 Navigation Pane

The navigation pane contains the “model view” or alternatively the “file view”. The model view is, by default, a tree view of the case/model. The file view display all files currently used by OLGA.

By default, the model view uses a tree-view to group the different components in the model. Selecting an entry in the tree-view will present the properties of the entry in the property editor. It is also possible to change the presentation to a flat hierarchy.

The groups and components are by default sorted by *type* but can also be optionally sorted alphabetically. It is also possible to toggle if the navigation pane should include all cases belonging to the project.

8.2.5.2 Property Editor

The properties are by default sorted according to “used keys” other alternatives are alphabetical order, “complete” and according to state.

Used keys arrange the properties in groups and include a “not used” grouping where some properties are placed. It is unclear if the properties grouped under “not used” are irrelevant to the case/model or simply not set by the user. The used keys sorting include behavior where some properties are moved around when a value is set or removed.

Alphabetical order replaces all groupings with the group “alphabetic” and arranges all properties alphabetically in this group.

Complete order the properties in groups similar to *used keys*, but does not include a “not used” group. Instead, all properties have a static position in their respective groups.

The **state** order replaces the groupings with two groupings, “your selection” and “not used”, properties without a value are placed in “not used” and are moved to “your selection” when a value has been set.

Located next to the buttons for the different sorting orders are buttons for “Property page” and “Timeseries” which launches separate interfaces for *some* of the components in the model. The buttons are disabled if the currently selected component does not support the functionality.

Between the previously mentioned buttons and the property list is a dropdown menu indicating which object/component the properties belong to. The dropdown menu can be used to navigate the objects/components at the same level in the tree view, but cannot be used to navigate out or into sub-components.

For example, a flow-path may include sub-objects such as piping and output. When the current *active* component is a node, the dropdown list can be used to navigate to the flow-path, but cannot be used to navigate to flow-path → piping or flow-path → output.

Properties are presented in a two-column table list with the property labels in one column, and the text/numerical input control in the second. Some properties have a dropdown list instead of text input, and some properties require comma separated values.

8.2.5.3 Handling the Model Browser

The properties displayed are for the currently selected component in the diagram view. It is possible to select another component or entry in the model view, which will select the corresponding component in the diagram view.

To use the model browser effectively the model view must be used in concert with the property editor. This require an amount of the visible work area, and due to the floating nature of the model browser, it may partially cover the model, or be in the way when working on larger models. It is possible to hide the navigation pane and use the diagram view to select components, but it will not be possible to reach all property settings. It is also possible to minimize the model browser, but this may not always be practical when building a model.

The property editor use three different font colors to convey if a property is required, optional or not used, the user manual describes the colors as seen in Table 8-2 (*User Manual OLGA 7.3*, n.d).

Color	Description
Black	Property can be given but not required
Red	Property required
Gray	Property will not be used

Table 8-2: Description of text colors used in the property editor

The use of colors and the terminology used in the documentation regarding the colors are unclear and do not match with the observed behavior, unless they are only valid when properties do not have a value set.

Red properties (required) are only made red when the property value is empty, and are black when they contain a value.

Black properties can turn red or gray when the value field is cleared.

Gray properties can turn black when a value has been set. It is also unclear if gray properties are optional or if they will be ignored even if a value has been set. Some of the gray properties will remain gray but accept input, and some refuse input.

The “not used” grouping is also increase the confusion the group may contain black properties along with gray properties.

In some situations, setting the value of an empty black property, may result in the simulator state displaying “not Runnable” and performing a verify action from the toolbar does not give any indication of where the issue is located.

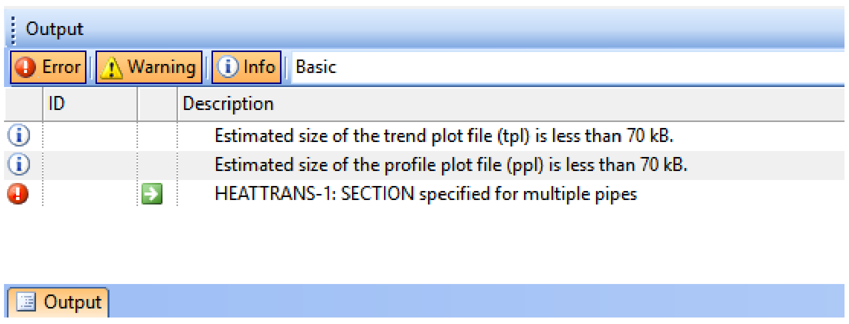
On invalid data-input, the behavior removes control and silently modifies the value, which may not be spotted by the user. For example, entering text in a property requiring a numerical value will silently change the entered text to 0 (zero) when the input control lose focus.

Some input controls also include a dropdown list with available units. The entered value is converted to the new unit when a unit is changed.

8.2.6 Output Tab

The output tab (Figure 8-14) displays messages from the GUI, and from interactive simulations, the output tab can also display messages from other active cases, by selecting the desired case from a dropdown menu. The message types displayed are errors, warnings and informational messages, critical messages are shown with popup dialog boxes.

The output entries are presented in a list with columns with an icon column indicating type of message, an ID column, an unlabeled column for buttons, and a description column. The buttons appearing in the unlabeled column will navigate the user directly to the location of the issue. In some situations, invalid settings in the case definition are not registered as errors. The error will not appear in the output, though the simulator state will appear as “not Runnable”.



	ID		Description
ⓘ			Estimated size of the trend plot file (tpl) is less than 70 kB.
ⓘ			Estimated size of the profile plot file (ppl) is less than 70 kB.
❗		➡	HEATTRANS-1: SECTION specified for multiple pipes

Figure 8-14: output tab with informational and error messages

8.3 Plot View

The UI varies depending on the plotting module used, thus the interfaces for interactive and static plots are discussed separately. As previously mentioned, the terms interactive and static, are used to separate the two different modules of the plotting functionality integrated in OLGA. The prefix *Interactive* is actively used in the OLGA interface and documentation. *Static* is only prefixed to keep the two modes separated; the phrasing is not intended to imply that the referred plot mode is locked (*i.e.* unchangeable). The phrasing is better fit with where the different modes retrieve the simulation data. The integrated plotting functionality in OLGA is provided by the TeeChart software; both modules provide more or less the same functionality and configuration options, with some differences.

8.3.1 Interactive Plots

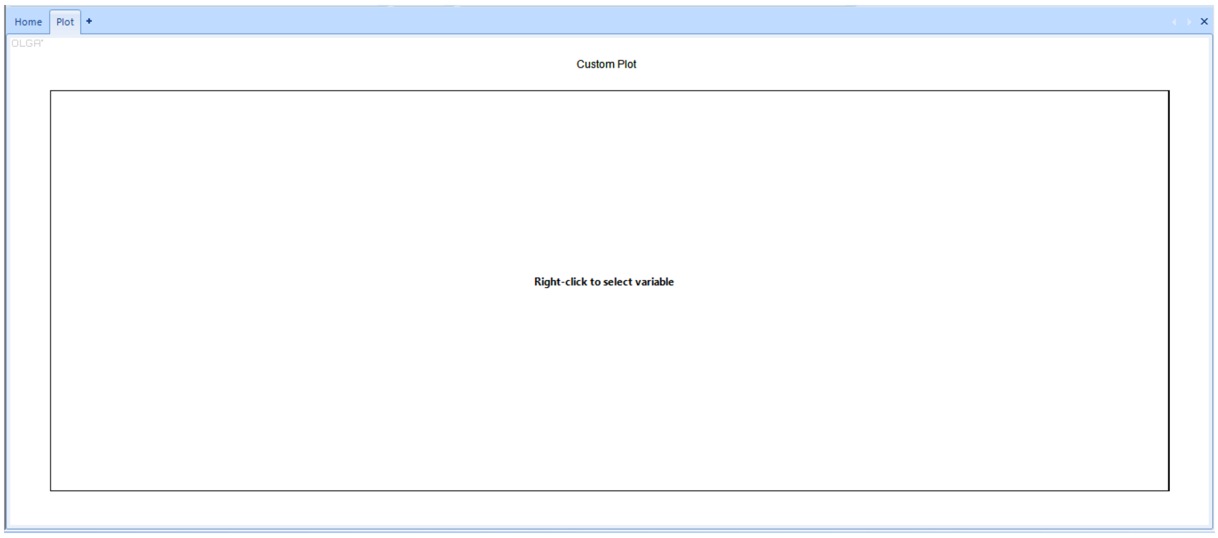


Figure 8-15: interface for interactive plots

As can be seen in Figure 8-15 the interface relevant to the plot is minimalistic, providing only a title and a canvas with instruction on what to do to add variables. The main portion of interaction goes through the context menu, resulting in few distractions in the interface.

Selecting a variable to plot is straightforward, right click → Edit/select Variable → tick desired variable → click "OK". A variable is now specified for the canvas, nothing is plotted and no more instructions for plotting the data are provided. Instructions are available in the OLGA manual, but this require that the user realize that the function referred to as "Multiple plots" in the toolbar tooltip, or as "Custom plot" in the title, one location in the manual, and among users, are referred to as interactive plots in the manual.

8.3.1.1 Context Menu

The context menu for interactive plots (Figure 8-16) consists of the following entries:

- Edit/Select Variable
- Remove All Variables
- Show variable selector
- Max/Min Settings
- Edit X-axis Unit
- Show Border (only visible with multiple canvases)
- Load Layout from File
- Save Layout to File
- Add Plot
- Remove Plot
- Edit Title
- Layout
  - Maximize
  - Minimize
  - Normalize
  - Flip
- Copy
  - Image
  - Data
  - All Images
- Configuration
- View
  - Plot
  - Value
  - Post-processed 3D plot
  - Post-processed plot

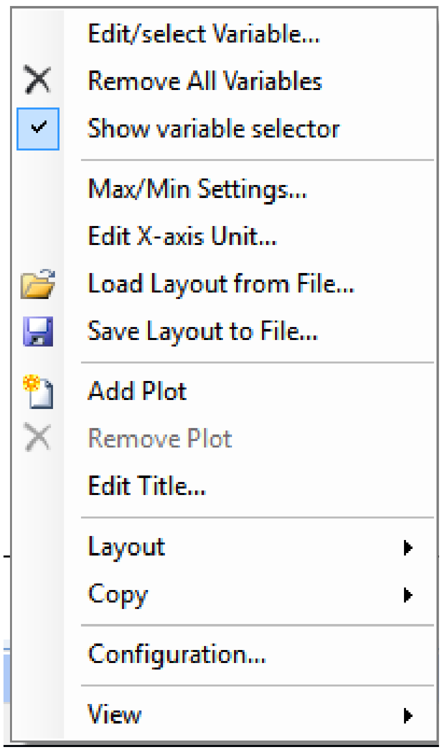


Figure 8-16: image of context menu

8.3.1.2 Grid

Interactive plots enable the creation of multiple canvases in a grid (Figure 8-17), a canvas can contain a trend/profile plot, the value of a variable, a post processed 3D plot, or a post-processed plot. New canvases can be added by using the **Add plot** action which will present a four button arrows pointing up, down, left and right as well a red X button indicating abort, new plots are added relative to the canvas where the context menu was activated. To remove a plot the context menu must be activated in the canvas to be removed.

The grid layout can be modified using the entries under layout, which can maximize or minimize the selected canvas (where the right click is registered) these actions will resize the canvas so it either takes most of the space or as little space as possible.

The normalize action resizes all canvases in the grid, so they take the same space, given the example shown in Figure 8-17 the canvases would be resized so the height of the two "rows" are equal, and resize to equal width on the two canvases on the bottom.

Using the flip action would flip the layout, and in the example below if the context menu were activated on the canvas at the top it would change the position of the two rows, while if activated on one of the lower canvases it would flip the location of the two horizontally.

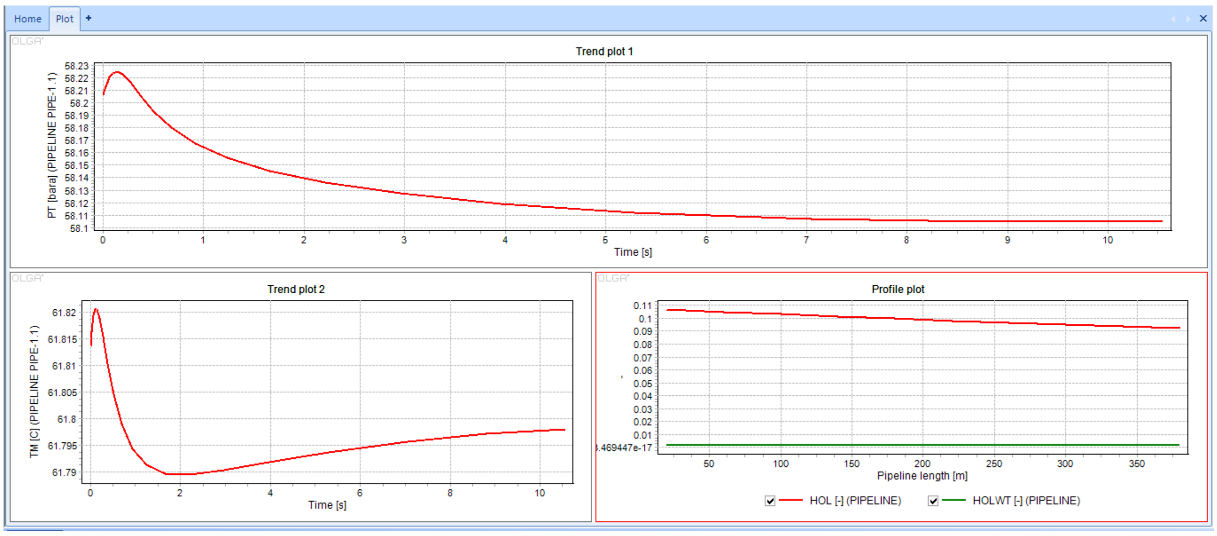


Figure 8-17: interactive plot with charts arranged in a grid

Max/Min Settings opens a window containing settings for manually setting maximum and minimum ranges on the X- and Y-axes. Settings for "Horizontal axis sliding window" are also present but are disabled with no obvious method of making them active. Changes made can be applied using the “OK” or “Apply” button, or unapplied changes can be cancelled using “cancel”.

Edit axis unit opens a new window containing only a dropdown list and a “Close” button. Similarly, the **edit title** entry only consist of a text input and buttons for “OK” and “Cancel”.

Load layout from file can be used to load a previously created grid setup along with variable selections. Moreover, **save layout** can be used to store a created grid along with the variable selections to a file. Plotted results are not stored in the layout file.

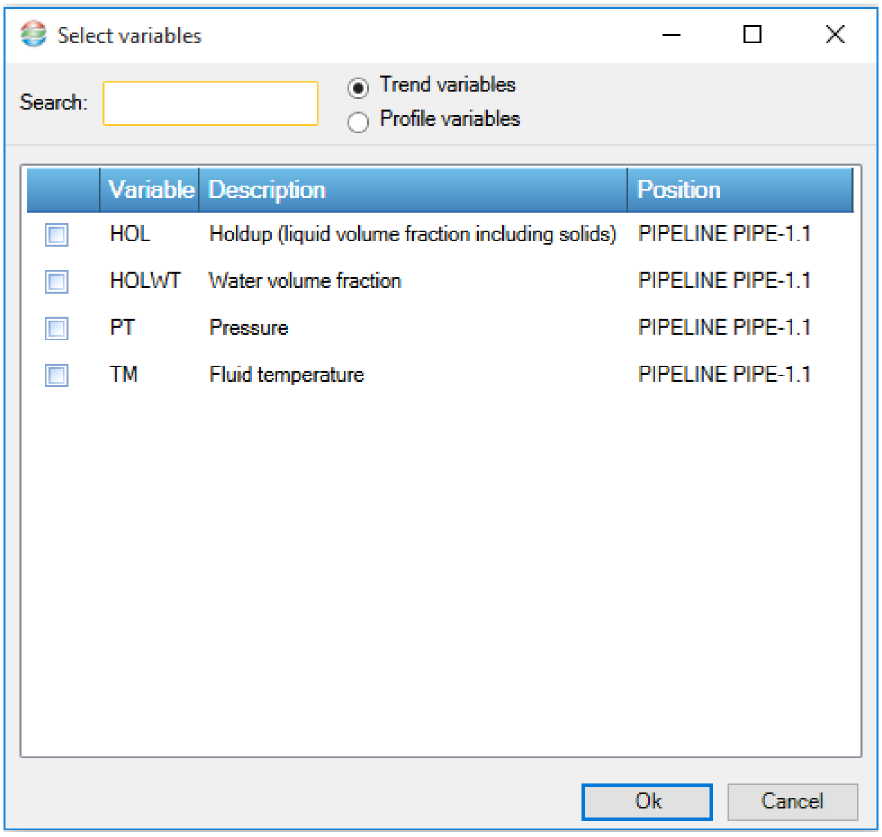
Under **copy,** it is possible to copy an image of the current canvas, or all the data from the selected canvas. It is also possible to copy all the canvases in a single image using **all images**. The all images entry may in some situations appear twice in the menu, where one of the “all images” entries does not perform any action.

The **show variable selector** entry does not appear to have any function, and **show border** can be toggled to either show or hide a border around the selected canvas.

8.3.1.3 Selecting Variables

Edit/Select Variable (Figure 8-18) opens a new window with all available variables in a list, it is possible to switch between trend and profile variables using radio buttons (trend/profile variables cannot be mixed). A search field is included and searches are performed on all columns in the list. Variables can be selected by activating the corresponding checkbox. The selection can be applied by clicking “OK”. Clicking cancel will not make any changes to the plot, except canvases used to view the *value* of a variable, which is cleared regardless of the action performed by the user.

The variable selector imposes some restrictions on how many variables can be plotted to the same chart. Only variables with *compatible* values can be plotted to a chart, incompatible variables are automatically hidden.



[✓ Trend variables] Profile variables

	Variable	Description	Position
☐	HOL	Holdup (liquid volume fraction including solids)	PIPELINE PIPE-1.1
☐	HOLWT	Water volume fraction	PIPELINE PIPE-1.1
☐	PT	Pressure	PIPELINE PIPE-1.1
☐	TM	Fluid temperature	PIPELINE PIPE-1.1

Figure 8-18: variable selector dialog for interactive plots

8.3.1.4 Configuration Window

Configuration opens a new window with the configuration options (Figure 8-19). The configuration options are arranged in tabs, and nested tabs are heavily used, which renders it difficult to navigate or perform desired changes. Various functionality such as changing colors and labels and much more are available in the configuration window, but is difficult to locate or even predict where is located. A lot of apparently unnecessary functionality is also provided, adding a lot of distractions and complexity. Finally, the window size is by default not large enough to show all available tabs horizontally, and some configuration pages do not have enough height to display all options. The window can be resized, but the custom size is not remembered when the configuration window is opened later.

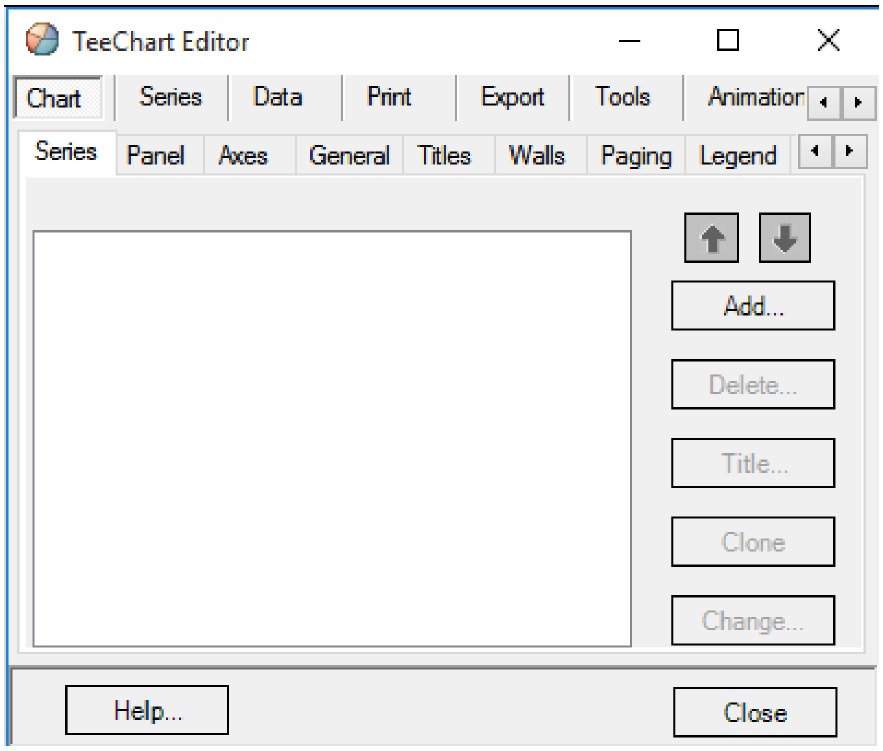
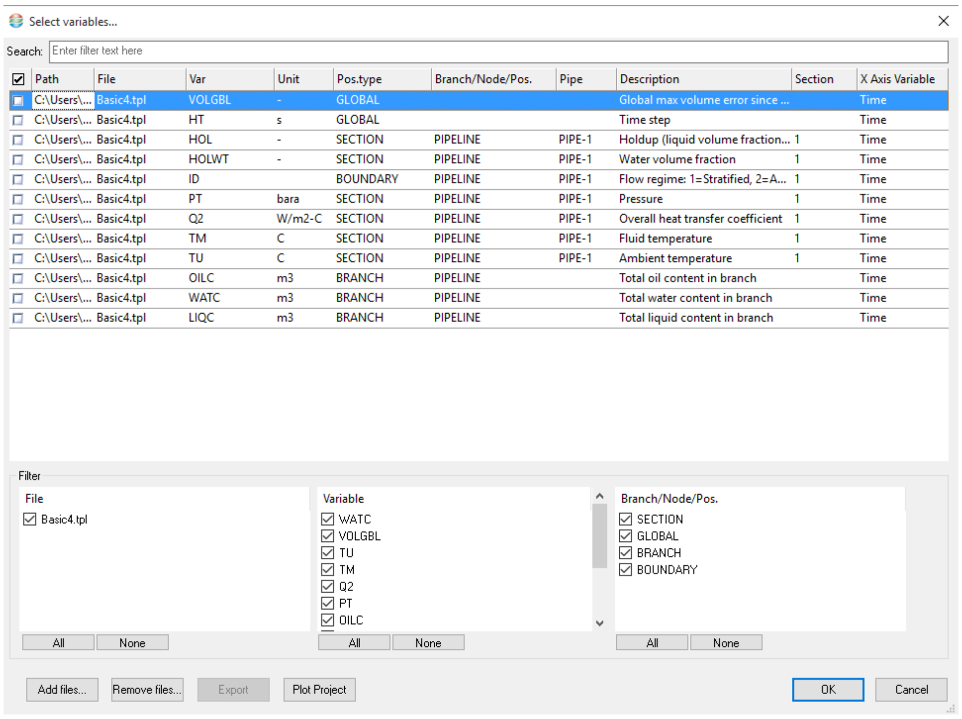


Figure 8-19: configuration window for interactive plots

8.3.2 Static Plots

To create a “static” plot the toolbar buttons for “Trend plot” or “Profile plot” can be used. When a static plot is to be created, the user is presented with a dialog with available variables to plot (Figure 8-20). The dialog includes a search feature, a column list of variables, and filter options for file, variable, and the branch/node/position. At the bottom are buttons for adding and removing result files, exporting the data of *selected* variables, a button for quickly adding all result files associated with the current *project*, and buttons for creating the plot or abort plot creation.



Select variables...

Search: Enter filter text here

☑	Path	File	Var	Unit	Pos.type	Branch/Node/Pos.	Pipe	Description	Section	X Axis Variable
☐	C:\Users\...	Basic4.tpl	VOLGBL	-	GLOBAL			Global max volume error since ...		Time
☐	C:\Users\...	Basic4.tpl	HT	s	GLOBAL			Time step		Time
☐	C:\Users\...	Basic4.tpl	HOL	-	SECTION	PIPELINE	PIPE-1	Holdup (liquid volume fraction...	1	Time
☐	C:\Users\...	Basic4.tpl	HOLWT	-	SECTION	PIPELINE	PIPE-1	Water volume fraction	1	Time
☐	C:\Users\...	Basic4.tpl	ID		BOUNDARY	PIPELINE	PIPE-1	Flow regime: 1=Stratified, 2=A...	1	Time
☐	C:\Users\...	Basic4.tpl	PT	bara	SECTION	PIPELINE	PIPE-1	Pressure	1	Time
☐	C:\Users\...	Basic4.tpl	Q2	W/m2-C	SECTION	PIPELINE	PIPE-1	Overall heat transfer coefficient	1	Time
☐	C:\Users\...	Basic4.tpl	TM	C	SECTION	PIPELINE	PIPE-1	Fluid temperature	1	Time
☐	C:\Users\...	Basic4.tpl	TU	C	SECTION	PIPELINE	PIPE-1	Ambient temperature	1	Time
☐	C:\Users\...	Basic4.tpl	OILC	m3	BRANCH	PIPELINE		Total oil content in branch		Time
☐	C:\Users\...	Basic4.tpl	WATC	m3	BRANCH	PIPELINE		Total water content in branch		Time
☐	C:\Users\...	Basic4.tpl	LIQC	m3	BRANCH	PIPELINE		Total liquid content in branch		Time

Filter

File	Variable	Branch/Node/Pos.
☑ Basic4.tpl	☑ WATC	☑ SECTION
	☑ VOLGBL	☑ GLOBAL
	☑ TU	☑ BRANCH
	☑ TM	☑ BOUNDARY
	☑ Q2	
	☑ PT	
	☑ OILC	

Figure 8-20: variable selector for static plots

The **search feature** performs a search on all elements in the list and columns with the result that when entering text in the search field will attempt to match the input to the content in all contents. This means that the user cannot specify if they are searching for a variable name, unit, or description text. In some situations, the search term may not filter any items, for example given a case with a name containing "tu" and using search to filter out all variables except the variable "TU" will not filter out anything.

The **list of variables** lists all simulated variables, it is possible to highlight multiple variables and toggle the selection state with the keyboard space-key. Include columns for *selected status*, *Path*, *File*, *Var*, *Unit*, *Pos.type*, *Branch/Node/Pos., Pipe, Section, X Axis Variable* and *description*. Each column can be used to sort the list. When clicking on the column label for any of these columns will group the listed variables according to the sort pattern, the groupings can be sorted alphabetically or reversed alphabetically, the order of the variables inside the groups cannot be modified. The sorting behavior may appear to be different between the various column options, when working with one result file in one path. Some column sorting may appear to be one or two expandable groupings containing the variables (as the example in Figure 8-20), or appear with one expandable grouping for *each* variable.

Each column label groups the variables differently:

Selected status – Groups all variables in a selected group

Path – Groups variables according to the path where the file is stored
File – Groups variables according to the filename containing the variable
Var – Groups variables according to variable name
Unit – Groups variables according to the unit
Pos.type – Groups variables according to the position type
Branch/node/pos. – Groups variables according to the column value
Pipe – Groups variables according to the pipe
Section – Groups variables according to the section number
X Axis variable – Groups according to the column value
Description – Groups variables according to the description

Since each sort option groups the listed variables according to their specific column-values, the list may appear much more crowded when ordering by certain columns. For example with only one result file, ordering by the *var* column will result in a list with one group for each variable, the grouping issue is similar when ordering by description. The grouping issue in the example is significantly improved when including variables from multiple files, which will group multiple variables in one group.

Filter options can be used to show/hide elements in the variable list, the *file* filter can be used to toggle the visibility of variables from specific files. The *variable* filter can be used to toggle the visibility of specific variables. The filter for branch/node/pos. can be used to toggle the visibility of variables in specific branches/nodes/positions.

For each filter pane, there are two buttons “All” and “None” which can be used to select all or deselect all. It is not possible to highlight multiple filter options and toggle the selection state (with space) each filter must be toggled one by one or by using the all or none buttons.

When variable selection has been completed the chart will be created, the interface and chart layout differ from interactive plots.

The differences are:

Static plot variable selection does not restrict the amount of variables plotted to a chart, compatible and incompatible variables can be plotted to the same chart, and the only restriction is that profile variables and trend variables cannot be plotted in the same chart.

A static plot includes a toolbar for some actions, while interactive plots do not.

Static plots cannot arrange multiple plots in a grid.

In static plots the legend is displayed above the chart, in interactive plots the legend is located below the chart.

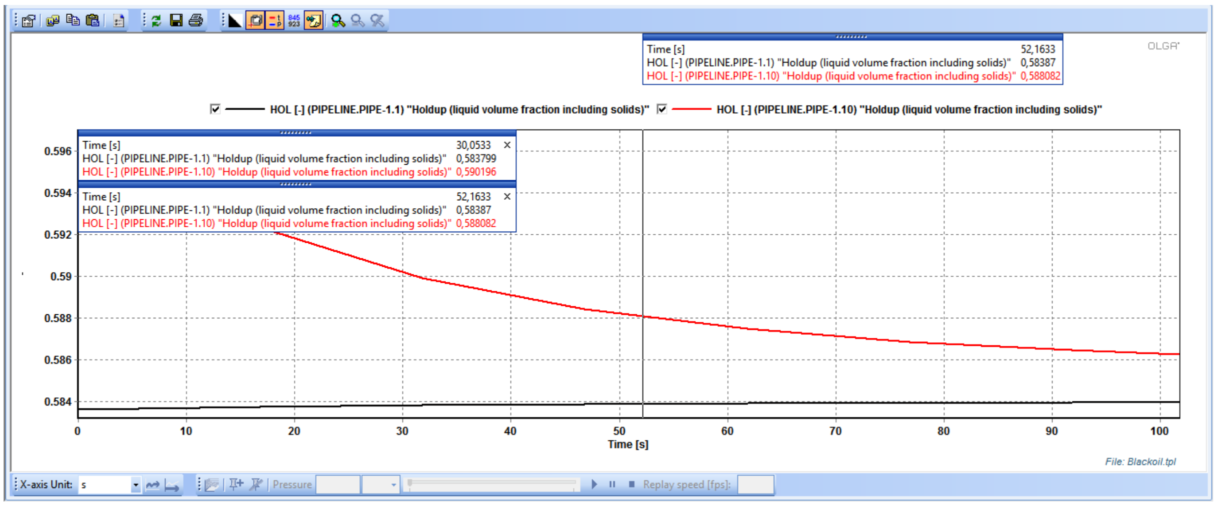
Static plots do not include a title by default, whereas interactive plots include the title “custom plot”.

8.3.2.1 Plot Toolbar

A static plot includes a toolbar with buttons for displaying the variable selection dialog, copy the chart as an image, copy the data used in the plot, paste data, display the plot configuration dialog, reloading the chart, save as image, print, display chart in black and white, toggle axes collapsing, toggle legend visibility, track values, toggle notes visibility, zoom functionality.

The **paste data** functionality on the toolbar is unclear how to use, or if it is at all useful, it does not have any obvious explanation for use, and provide errors when attempting to paste data.

The *track values* functionality can be very useful when the values of specific points are needed. When the functionality is enabled the cursor can be hovered over points in the chart, a small window with the details for the hovered point will follow the cursor, a left click will add a “sticky” window on top of the chart, enabling making multiple small windows with details from different points in the chart (Figure 8-21). The functionality can be disabled with a mouse right click or by toggling the toolbar button; “sticky” windows are not removed when value tracking is disabled. This functionality may present some precision errors. The toolbar button does not indicate if the functionality is enabled but a vertical line and a window with details are visible in the chart when the functionality is enabled. The issue with the button not indicating the status is that it is not immediately obvious that the functionality can be disabled by toggling the button.


Time [s] 52,1633
HOL [-] (PIPELINE.PIPE-1.1) "Holdup (liquid volume fraction including solids)" 0,58387
HOL [-] (PIPELINE.PIPE-1.10) "Holdup (liquid volume fraction including solids)" 0,588082

Time [s] 30,0533
HOL [-] (PIPELINE.PIPE-1.1) "Holdup (liquid volume fraction including solids)" 0,583799
HOL [-] (PIPELINE.PIPE-1.10) "Holdup (liquid volume fraction including solids)" 0,590196

Time [s] 52,1633
HOL [-] (PIPELINE.PIPE-1.1) "Holdup (liquid volume fraction including solids)" 0,58387
HOL [-] (PIPELINE.PIPE-1.10) "Holdup (liquid volume fraction including solids)" 0,588082
OLGA
HOL [-] (PIPELINE.PIPE-1.1) "Holdup (liquid volume fraction including solids)"
HOL [-] (PIPELINE.PIPE-1.10) "Holdup (liquid volume fraction including solids)"
0.596
0.594
0.592
0.59
0.588
0.586
0.584
0
10
20
30
40
50
60
70
80
90
100
Time [s]
File: Blackoil.tpl
X-axis Unit:
s
Pressure
Replay speed [fps]:

Figure 8-21: static plot with value tracking enabled

8.3.2.2 Control Tools

Below the chart is a control bar (Figure 8-22), with most functionality only enabled when working with profile plots. The only functionality enabled in trend and profile plots, is the X-axis unit selection. Other functionality (listed below) that is located on the control bar is only enabled for profile plots.

- Button to use pipeline length on X-axis
- Button to use horizontal length on X-axis (will deactivate above button when activated)
- Button to display the chart as a 3D-surface
- Button to "keep graph", retains the plotted graph at specified time step and make it possible to plot the same variable at multiple time steps
- Button to remove graph, removes all plotted graphs for different time steps
- Time field displaying the time step the current graph is for
- Time unit dropdown selection, allowing the user to select between [s]econds, [M]inutes, [h]ours, [d]ays and 1/rpm.
- Playback seek-bar
- Playback controls (play, pause, stop)
- Replay speed field

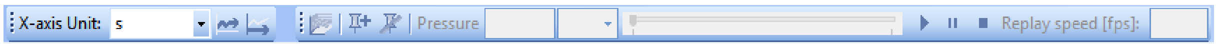


Figure 8-22: control tools with functionality mainly for profile plot

8.3.2.3 Chart

The chart created with static plotting is similar to charts created with interactive plotting, the differences are:

- Only one chart can be created in a tab
- The results are taken from files containing the simulation results
- The title is by default hidden
- There are no restrictions on the amount of variables plotted
- The legend is located above the chart
- Static charts can contain multiple Y-axes
- Playback controls (for profile plots)
- Different structure on context menu

8.3.2.4 Context Menu

The context menu is activated with a right click on the canvas for the chart; all options are visible in both trend and profile plots. The structure of the context menu organizes the actions in three sub-menus, and differs from the layout used in interactive plots. The structure is as follows:

- File
    - Reload
    - Open template…
    - Save As Image…
    - Save As template…
    - Save As Video Clip
    - Print Setup…
- Edit
    - Select…
    - Copy
        - Copy Data
        - Copy Image
        - Settings
    - Paste Data

    - Note(s)
        - Add…
        - Edit… (disabled unless context menu is activated on a note)
    - Min/Max values…
        - Vertical Axis…
        - Horizontal Axis…
    - Series…
    - Legend…
    - Axis…
    - Titles…
    - Slug Statistics…
    - Surge Volumes…
    - Configuration…
- View
    - Black/White
    - Collapse Axes
    - Legend
    - Track Values
    - Pipeline length (only enabled in profile plot)
    - Horizontal length (only enabled in profile plot)
    - Notes

The sub-menu *View* contains toggle-able options that are also available on the toolbar. The sub-menu *Edit* contains functionality that can modify aspects of the chart, and the sub-menu *File* contains functionality aimed at handling files, printing and refreshing the chart.

The functionality for saving video clips can only be used with profile plots. However, the entry is not disabled when trend plots are used. Thus, the menu item can be activated but there will be no feedback.

In the view submenu, the functionality for pasting data does not have any obvious use case or indication of how/when the functionality can be used. The edit entry located under the sub-menu notes is only activated when the context menu is activated *on* a note.

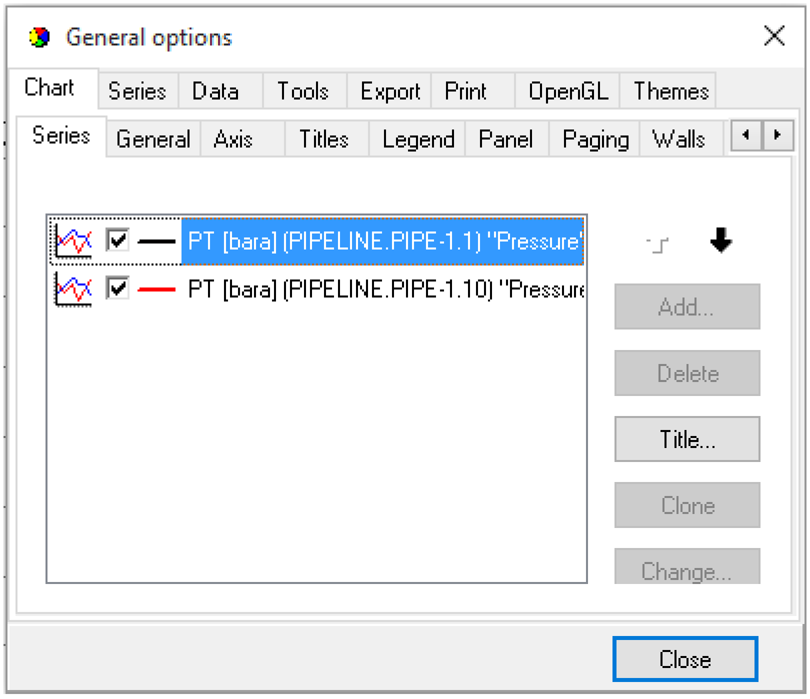
The **select** entry launch the variable selection window previously described in 8.3.2 Static Plots.

The **copy** submenu contains functionality for copying *all* the data for the plotted variables, copying an image of the plotted chart, and a *settings* entry that launch a dialog where the resolution of the copied image can be defined. The settings dialog contains checkboxes for using the plot window size (current resolution of the plot canvas), a checkbox for retaining the current aspect ratio, and two input fields for the desired width and height. If the checkbox for using the plot window size is selected then the remaining options are disabled.

The **add note** functionality launch a dialog consisting of a text area for the note and a dropdown menu where a specific variable to affix the note can be selected, as well as a cancel and ok button. The **edit** functionality is identical but with an addition of a delete button.

8.3.2.5 Configuration Window

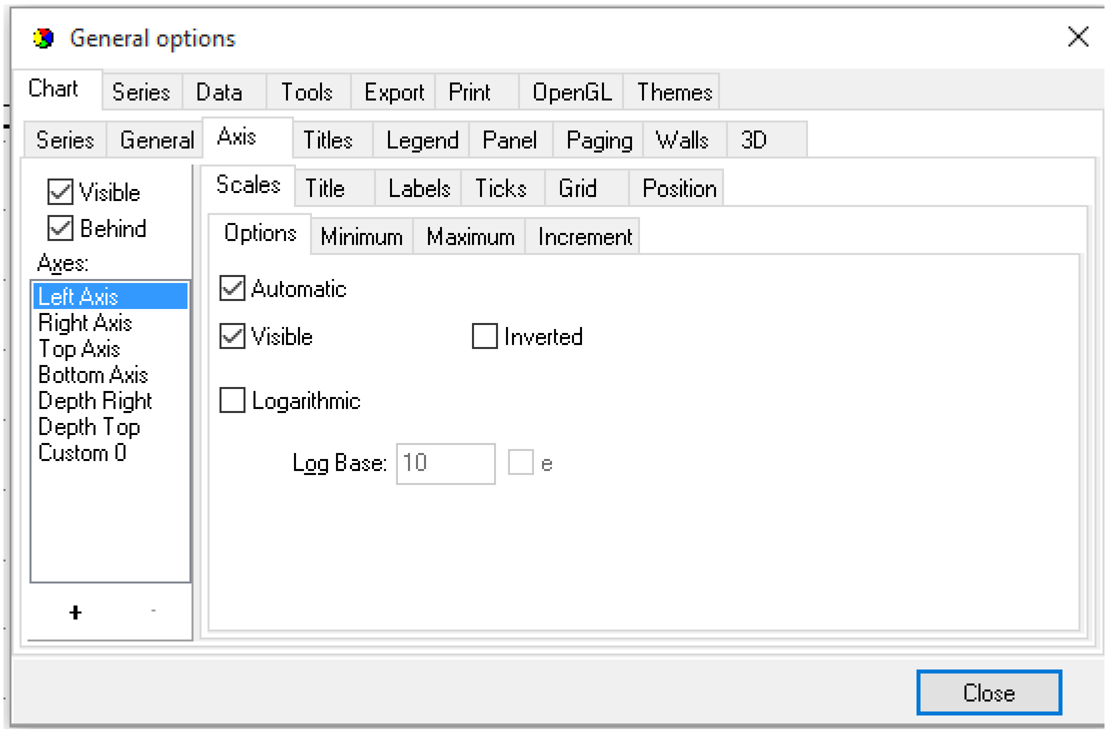
The **configuration** dialog is similar to the configuration dialog for interactive plots. However, the window size for static plots' configuration window is remembered between application launches. The aesthetic design of static plots' configuration is also (subjectively) more visually pleasing (Figure 8-23).



✓ PT [bara] (PIPELINE.PIPE-1.1) "Pressure"
✓ PT [bara] (PIPELINE.PIPE-1.10) "Pressure

Figure 8-23: configuration window used in static plots

The configuration windows for both interactive and static plot share the same issue with heavy use of nested tabs (Figure 8-24). In some areas, the nesting consists of up to seven tab levels. The heavy use of nesting, and deep nesting in some areas, make it more difficult to navigate, predict and locate functionality, and effectively overwhelms the user.



[✓] Visible
[✓] Behind

[✓] Automatic
[✓] Visible
[ ] Inverted
[ ] Logarithmic
Log Base: 10 [ ] e

Figure 8-24: illustration of tab nesting used in the configuration window

8.3.2.5.1 Custom Dialogs

The dialogs described in this section are custom dialogs, specific to OLGA, created to make it easier to complete certain tasks, without using the configuration window. The custom dialogs were developed after querying existing users of what functionality, from the configuration window, they used and needed. Without these dialog the user would be required to interact with the default configuration dialog of TeeChart.

Series… launch a dialog consisting of a dropdown list of plotted variables, a title section with radio buttons for using the default or user defined and a text input for the title, the input is disabled when the default radio button is activated. The title is used in the legend for the specified variable. The dialog also includes a dropdown list for line color selection and a checkbox to assign the color on the variable’s axis. A dropdown list for different line styles is also included, along with spin controls for the width and spacing of the line style.

Legend… launches a dialog with a checkbox toggling the visibility of the legend, as well as font selection, font size and location of the legend (options: top, bottom, right).

Axis… launch a dialog consisting of a dropdown list of axes, the list include both X- and Y-axis variables. Other elements include radio buttons for the position of the axis (Y-axis: right/left; X-axis: bottom/top), radio buttons for the axis label (default/user defined) and an associated text input (disabled with 'default'). Radio buttons for the "label format" is also included where the options are default or user defined. A dropdown list is enabled when label format's user defined is selected, the selected entry can be edited but any changes are discarded, when apply is changed the radio button selection returns to 'default' but keep the chosen format in the list. Font and color settings for the selected axis can also be specified. The exact rules for what options can be used and when they can be used is unclear, in some charts/variable selections, the options are all disabled, and in others, they are all enabled.

Titles… launch a dialog consisting of two groupings "header" and "footer". Both groupings include a checkbox for "visibility" a button for font configuration along with a non-interactive text field for current font and font size. The header grouping also includes a text input where the heading title can be defined. By default, the visibility checkbox of header is unselected.

The dialog boxes for the menu items: series, legend, axis and titles – include an apply button and a close button. When changes are made in the dialog and close button is used, a prompt appears and ask if the user want to apply the changes. The prompt present three options: yes, no, cancel – the yes and no options leads to the expected result, in the context of the prompt, the cancel button *should* abort the close action but will in reality serve the same function as selecting no.

Slug statistics… launch a dialog with a description of variables used to calculate and some details of how the functionality works. The available input for users are input fields for interval length, start time, and end time each input field has a dropdown list of time units (seconds, minutes, hours, days, 1/rpm) to select from. Action buttons available are reset, ok, cancel, apply.

Surge volumes… launch a dialog with four groupings: surge type, description, calculation time span, and maximum drainage rate. As well as action buttons: reset, ok, cancel, apply.

The surge type grouping consists of three radio buttons (liquid, oil, water) the active radio button modifies the content in the description grouping. The description grouping consists of the description and a checkbox for showing details, which add more details to the description.

Calculation time span include two text inputs with associated dropdown lists for time units, for defining when to start and end the calculation, if left empty the simulation start and end times are used.

Maximum drainage rate include a dropdown list for position a text input for Qmax with an associated dropdown list of units. The grouping also includes a list with the columns position, max surge volume, and QMax.

8.3.3 Exporting Data

To extract data results the users have a few alternatives, depending on the type of plot. To extract all results the context menus include an export function, which can be used to export all the results.

More commonly, is using the value tracking functionality to extract data manually from specific points in the chart. Value tracking requires that the user go through a procedure where (1) the data is visualized, (2) tracking used to specify data manually, (3) arrange the data in third-party tool, and (4) process the data. This is a procedure that is not only tedious and time consuming, but requires hand-eye coordination, cognitive attention, an overview of all the data and precision, including a risk of precision errors.

9 Methodology

The aim of this study was to discover possible usability issues in the GUI of OLGA. In addition, some accessibility aspects were considered.

As OLGA is used in a technical domain, and was completely new to the evaluator the process of gathering data was split into two different phases.

The first phases consisted of slightly modified contextual interviews where participants were interviewed and observed while using the application. The focus in the first phase was to gather data to assist in limiting the scope of the project. Multiple individual interviews were conducted in meeting rooms, to prevent interrupting others. During interviews, the screen of the interview subject was shared on a separate monitor, enabling face-to-face conversation while at the same time see what the participant was currently doing.

The second phase consisted of a focus group interview, where participants discussed the current state of the application. Based on observed trends in the previous phase with individual interviews the plotting functionality was selected as a focus point. However, participants were free to discuss other topics as well. Initially the focus group was intended to be multiple individual co-creation sessions. However, due to delays, the plan was modified and a group was organized. Each participant in the group was provided tools and encouraged to make personal notes or sketches during the group session.

9.1 Individual Interviews

Contextual interviews with the participants were conducted face-to-face in meeting rooms at their workplace. Interviews were semi-structured with a focus on allowing the participants to select the topic and present the OLGA GUI with how they use the application and what they conceived as issues.

The first interview was unstructured and was used as a guideline for all the later interviews. After each interview, the interview guideline was updated to add new topics. The length of interviews varied, based on the participants’ schedule and availability.

The participants were allowed to use their preferred version of OLGA, mainly the 2014 version and the (at the time) 2015 beta version.

9.1.1 Participants

Participants were provided by the contact at Schlumberger, who were asked to provide participants with varied experience with OLGA and with varied uses of the application.

Six participants were interviewed individually in the first phase, consisting of four consultants (2 females, 2 males) and two testers (males), the participants' experience with OLGA ranged from three years to approximately twenty years. The second phase consisted of a focus group with five participants, three consultants (2 females, 1 male) and two consultants (males).

Testers are not end users, but regularly work with the interface in order to discover bugs, errors and possible improvements. They were included as it was possible they would have valuable feedback, provide another view on interaction and, further, may have insight into why certain design decisions were made.

The user group of consultants is a group of users that uses OLGA in projects for external clients, such as analyzing how a planned production system may perform. Details about the production system is provided by the clients and the consultants transform the details into a case that is then fed to the simulator, and then the simulation results are handled, analyzed and the requested information reported to the client.

The consultants are also responsible for providing 1st-line support for OLGA and other Schlumberger software. More experienced consultants additionally perform reviews of other consultants' cases and simulation results as a form of quality assurance.

9.2 Protocol

Participants were recruited from Schlumberger by the advisors, who informed the participants about the project and interviews. Moreover, the interviewer informed the participants about the project and interview topic before interviews began.

The study did not focus on gathering personal information from the participants. However, privacy was still considered important. When audio recordings were used the participants signed agreement forms, and all audio recordings were encrypted and stored on a separate offline medium. Furthermore, when audio recordings were transcribed any personally identifiable and sensitive information was left out.

10 Results

This chapter lists the findings from the individual interviews, focus group and experiences and observations from frequent use of OLGA. Some of the findings from experiences and observations may not be relevant for versions of OLGA that are more recent.

10.1 Individual interviews

The individual interviews highlighted some issues with OLGA. The participants were allowed to use their preferred version of OLGA, which resulted in some issues that were introduced in more recent versions than the one the evaluator used (the evaluator used version 7.3.4 while participants varied between versions 2014 and 2015).

10.1.1 Issues

The issues are split into three groupings *general*, *modeling* and *plotting*.

10.1.1.1 General

Type	Issue
Stability	Unstable under heavy load
Stability	Tools may reduce stability
Flexibility	Unable to permanently close tool tabs
Flexibility	Unable to completely remove tool tabs
Documentation	Manual does not bridge gaps in knowledge
Documentation	Manual not updated and vague in some areas
Bug	Toolbar tooltips for plot toolbar disappears
Bug	Create-button on new-page sometimes require scrolling

Table 10-1: list of general issues with OLGA

Stability under load – Most of the data created and used by OLGA is stored in ASCII (text) that may require more space for storage and memory (RAM) to load. When a case with many result-files is loaded, OLGA will load all of the belonging files and the entire contents of the files are loaded into memory, which reduces stability when running out of memory. This may also affect the platform on which OLGA is running. The storage format also opens for, according to a participant, an additional possible issue, where a large case stored on the system disk and *something* goes wrong it may result in the entire disk space being used.

Tools reducing stability – This is an issue that was not further investigated, due to tools being considered separate functionality and lack of access to the tools. However, it is possible this issue is caused by resource starvation, as the participants reporting this usually worked with large and complex projects. It must also be noted that, reportedly, versions that are more recent have improved this issue.

Unable to permanently close tool tabs – Some tools, added as tabs, may be used to assist in building and defining a case, such as OLGA for Wells (O4W). The tool may be used to generate a model, but when that task is completed, the user may want to close the tab permanently, without losing the generated model or definitions. It was mentioned that it is possible to remove tools, but this required manually editing files, which was not demonstrated and should ideally not be necessary.

Unable to permanently remove tool tabs – Similar to the previous issue, but focused on completely removing unused tool tabs from the case, without manually editing files.

Manual not bridging gaps in knowledge – Participants with lower experience with OLGA reported that the documentation did not sufficiently assist in bridging gaps in knowledge. Examples given were in context of the description of properties, where the documentation would simply repeat the description. When a description is not enough, the documentation should provide a more detailed description and guidance on what to do. The **vague and out of date** manual was reported in context of the 2015 version of OLGA, which at that time was in beta version which likely means some of the issues were due to the active development of OLGA. However, while checking the manual provided with the older 7.3.4 version of OLGA illustrate similar issues, with some terminology appearing to be from a previous version of OLGA.

Furthermore, two issues, which are likely to be unintended bugs, were reported. In OLGA the plot tabs can be dragged so that the plot is displayed in a separate window, when this is done for static plots it results in the **tooltips no longer being displayed** when the pointer hover over buttons in the toolbar. Another, apparently unintended bug, is located in the new-page, a participant demonstrated that in some situations **scrolling is required to reach the create-button**, however, attempts to recreate the issue has been unsuccessful, as the new-page automatically adjust to the available space. The scrolling issue may be an edge case with unclear steps to reproduce, or it is limited to newer versions of OLGA.

10.1.1.2 Modeling Tool

Location	Type	Issue
Navigation pane	Effort	Navigating the tree view requires a lot of mouse clicks
Diagram view	Convenience and visibility	Visual layout of model not retained when partially copying
Model Browser		Large models partially covered by Model Browser
Fit to page		When used, model is partially covered by floating windows
Case tab	Affordance	No visible, or easy to reach, method to close a Case tab
Pipeline editor	Visibility	Difficult to determine which Case an instance belongs to

Table 10-2: list of issues in modeling tool

Navigation pane – The navigation pane rely on a tree view, with expandable branches, to display the components a model consists of. The tree view has the advantage of displaying hierarchies in a quick and intuitive way. A drawback of a tree view with expandable branches is that to expand a branch some effort must be exerted, some components are not immediately visible in the navigation, and buttons to expand a branch are usually small. In OLGA the navigation pane can be used in concert with the diagram view, selecting a component in the view will focus the entry in the navigation pane. However, not all entries in the navigation pane can be found in the model.

Diagram view – When a partial selection is copied to a different case, the visual layout of the components are not retained. The result is that the pasted components are sometimes randomly placed on top of each other, requiring that the user manually recreate the desired visual layout.

Model Browser – The model browser is a floating window, some participants reported that the window often is in the way and covering the model, requiring either frequently moving the window around or frequently minimizing and restoring the window. Some of the participants preferred a static pane for the model browser. Other participants preferred the current floating window, as this could be taken advantage of in multi-monitor setups.

Fit to page – Some participants reported that using the fit to page functionality did not perform as desired. The desired behavior was that zoom and pan would be used to fit the model in the visible space between the component view and model browser. The actual behavior is that zoom and pan place parts of the model under the floating windows.

Case tab – During interviews, the observed behavior of the participants was to open the file menu and use the close project button, when closing one case tab. The close project button is trivial and safe to use when only one case is open in OLGA. However, it is impractical to use when multiple cases are open and intending to close only one case tab. The toolbar includes a remove case button, hidden in a sub-menu under duplicate case, which can be used to remove cases from a project. The remove case button presents three options (previously discussed in 8.1 File Menu) all resulting in permanently removing a case from the current project, which may not always be desired. Participants were not asked if the remove case functionality was used, as the interviewer was not aware of the functionality, due to an assumption that the sub-menu of duplicate case would provide a similar functionality and not the opposite functionality (add vs. remove). Furthermore, when close project is used it only checks for changes to the project, if a Case has been edited and closed, with no changes to the project, it will not prompt to save changes in the Case. If the project has been modified, or if it is a temporary project, the user will be prompted and have the ability to save all modifications.

Pipeline editor – When many cases are opened in OLGA and frequently switching between case tabs, it is difficult to remember which case an instance of the pipeline editor belongs to, and thus difficult to predict which case that instance will generate a model for. The pipeline editor is only available in the 2014 and later versions of OLGA, and was not available to the evaluator. Some participants reported they avoided using pipeline editor as it was difficult to use, some reported stability issues, and one participant reported using Excel to prepare data input to import into the editor as this was easier, more efficient and stable.

10.1.1.3 Plotting

Location	Type	Issue
Chart	Visibility	Defaults to green line color when exceeding ten variables
		Low contrast between graph lines
		Low contrast between background and foreground (graph lines)
		Low/No contrast between same variable for different time steps
	Reliability	Graph lines disappear from the chart
Labels	Knowability	OLGA specific terminology used for variables and axis labels
Legend	Reliability	Variables not included in legend when exceeding ten variables
	Knowability	OLGA specific terminology used for legend labels
Y-axis	Visibility	Each new axis requires a lot of horizontal space
Configuration	Affordance	Difficult to discover how to change graph line colors
		Difficult to locate relevant functionality
	Cognition	Amount of tabs and tab nesting overwhelms users

Table 10-3: findings in plotting tool

Defaults to green – When variables are plotted, they are automatically assigned a color. When the amount of variables selected exceed ten, then up to ten variables are assigned individual colors while the rest is assigned a green color. This results in a chart where a large amount of the plotted variables cannot be identified, due to appearing identical to other variables. It is possible to assign colors manually, the functionality may however be difficult to locate.

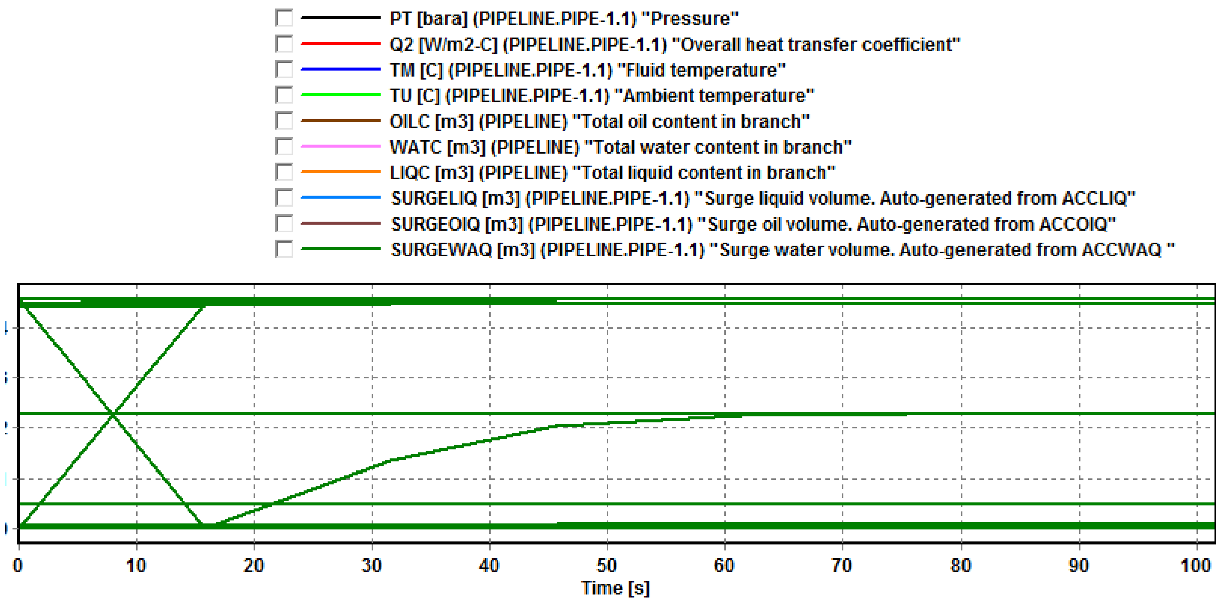


Figure 10-1: graph lines defaulted to green and incomplete legend

Low contrast between graph lines – With automatically assigned colors, results in some graph line combinations using combinations of green, red, blue, and so on, this can be problematic for colorblind users, of either OLGA or the exported chart. The color use is further problematic as the luminance and strength of the colors used are equal for many of the combinations, resulting in low contrast between the graph lines (see Figure 10-2).

Low contrast between foreground and background – The colors automatically assigned to variables, are also problematic with the background, where some colors can almost disappear into the background, as can be seen in Figure 10-2.

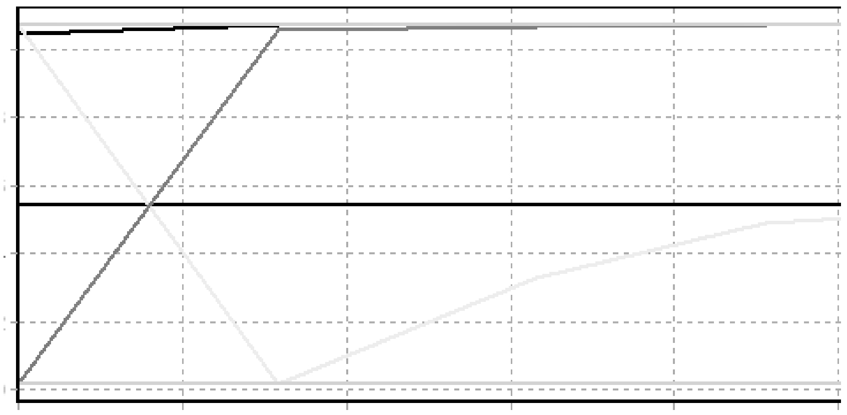


Figure 10-2: illustration of low contrast in plots (passed through image filter)

Low contrast between same variable of different time-steps – Profile plots are used to plot the same variable at different time-steps in one chart. When multiple time-steps are plotted the graph lines use the same base color (red for example) with slight differences in brightness, resulting in a visible difference for some users but not enough contrast for other users.

Graph lines disappear from the chart - When the amount of variables plotted reaches a certain amount, some graph lines are no longer visible. It is unclear if the specific graph lines are simply removed from the chart or if other graph lines hide the lines as illustrated in Figure 10-2.

OLGA specific terminology – The use of OLGA specific terminology is an issue since the charts may be shared with customers or other stakeholders who may not be familiar with OLGA or the terminology used in OLGA. The labels used on the axes use terminology specific to OLGA, most notably the Y-Axis labels. This is not an issue for experienced users of OLGA, but an issue for novices who may not be fluent in the various short-form names of variables, as well as non-users who may need or receive the plots from consultants. Furthermore, the same issue is apparent in the legend, as can be seen in Figure 10-3: it can be difficult to discern what the variable is.

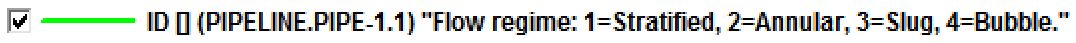

✓ ID [] (PIPELINE.PIPE-1.1) "Flow regime: 1=Stratified, 2=Annular, 3=Slug, 4=Bubble."

Figure 10-3: illustration of label used in the legend

The first two elements are of static sizes, while the short name, unit, location and description can vary in length. The descriptions can also vary in content, such as using a single word (ex. “Pressure”) or using a more descriptive sentence (ex. “Overall heat transfer coefficient”), or using a description which may need an extra description, such as “Surge liquid volume. Auto-generated from X” raising the question what is X and how does it affect the surge liquid volume.

Variables not included in legend – The amount of variables visible in the legend is limited to ten, if exceeding this amount some variables will not be visible in the legend (Figure 10-1), and cannot be hidden in the chart without deselecting the specific variable in the variable selector. It is unclear if the legend list the ten “first” variables (which is very likely), or the “last” ten, or another pattern without knowing the internal behavior of TeeChart.

Some of the issues previously mentioned, may only be relevant when plotting an excessive amount of variables in a chart, which is not practical or used in the *end result* by the participants. However, among some of the participants, it is a common practice to select all variables and plot them in one chart as a part of the workflow, mainly to see the behaviors to determine what is relevant. The workflow includes toggling the visibility of individual variables in the legend; Figure 10-1 illustrates how this workflow is broken by the stated issues.

Configuration window – As previously noted, the configuration window consists of many tabs and heavy tab nesting, which results in users being overwhelmed and the interface difficult to navigate, locate and remember where functionality is located. The complexity overwhelms users who are more likely to use third party software to perform the desired processing of charts, as an example, one of the participants mentioned using OLGA to create the chart, export as an image and import in paint to edit labels, despite OLGA including functionality to edit labels.

Difficulty in discovering how to change line colors – Functionality to change graph line colors are available in multiple locations in the configuration, the developers of OLGA has also created custom menu entries in the context menu with dialogs including this functionality. However, some participants struggled with locating the functionality, during one of the interviews the participant managed to locate the functionality for the first time, after being a user of OLGA for many years. This further illustrates the issues with the configuration window, and further, illuminates that the custom dialogs do not adequately advertise the available functionality.

Difficulty in discovering relevant functionality – As previously mentioned the complexity of the configuration window, result in difficulties when attempting to locate and remember where functionality is available. The participants did not use the custom menu entries during the interviews. However, this does not indicate whether the custom entries are used normally, which should be investigated when and how these custom entries are used, and if the users are aware of the functionality available in the dialogs.

Another way to view the issues related to discovering functionality is that the configuration window provides too much functionality to the user, thus requiring a more complex interface. Some of the functionality provided by the configuration is likely never going to be used, some functionality cannot be used as it breaks the plot and some functionality does not have any visible effect.

Y-axis use of space – When plotting many variables in a chart, not all variables will have compatible values resulting in new Y-axes being added. This creates an issue with available space for the chart as each Y-axis takes up an unnecessary amount of width space, as can be seen in Figure 10-1 and in Figure 10-4. Furthermore, the label can easily be visually grouped incorrectly, *i.e.* the label can be associated with the closest axis (on the left side), when it in reality belongs to the axis further away.

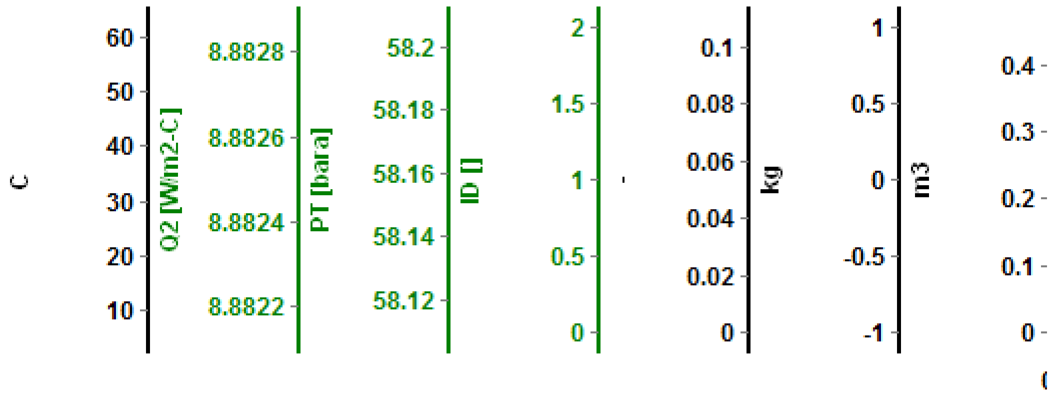


Figure 10-4: Cropped image of Y-axis labels

10.1.2 Requests and Suggestions

In this section, we discuss desires expressed during the interviews. The participants were encouraged to express their own desires, and made use of the opportunity to express desired functionality and behavior.

Two issues were also supplemented with descriptions for the desired behavior. The desired behavior of Fit to page was, according to one participant, that the action apply zoom and pan to make the whole model fit in the visible in the space, and not covered by the floating windows. The procedure of closing case tabs was also supplemented with a desired change in the UI, specifically adding a close button on the tab itself.

Table 10-4, below lists some of the suggestions from the participants.

Location	Suggestion	Type
Variable selector	Add apply button	UI change, flexibility
Documentation	More descriptive documentation	Documentation, support
Plotting	Different graph presets (ex. MATLAB)	Aesthetics, accessibility
	Support exporting EPS image format	Flexibility
	Simplify setting min/max axis values (multiple axes)	UI change, efficiency, effort
	Ability to add more variables after completing simulations	Flexibility, efficiency
Variable Selector	Increase default space for description column	Documentation, Visibility
Profile Plots	Only render static variables once when plotting multiple time-steps	Clutter
	Automatically add timestamp	Efficiency, identification
P&ID (symbols)	Add color codes	Identification
P&ID (diagram)	Ability to colorize diagram components	Flexibility, identification

Table 10-4: overview of suggestions

Apply button – The current variable selector provide two action buttons, for performing or aborting the current choices, both buttons will close the variable selector window. The apply button was suggested, based on situations where a participant was unsure what variables to plot. The situations often resulted in the participant open the variable selector, select/deselect variables and click OK to see the effects of the choices, a procedure often repeated multiple times. It was argued that the addition of an apply button would reduce the effort required when trying different combinations.

Descriptive documentation – One participant described OLGA as a software program that is easy to use, *i.e.* using the UI. However, the difficulty was in using domain knowledge in combination with OLGA and produce usable work. It was suggested that a more descriptive, and complete, documentation could assist in filling the gaps in knowledge. This suggestion was given in context of the 2015 version, where incomplete documentation could be defended on the, at the time, beta status. However, some of the issues mentioned, such as documentation repeating descriptions, were also valid in older stable versions of OLGA.

Graph presets – Different graph presets was suggested by a participant in the context of too low contrast in plots. The participant suggested MATLAB graph presets as examples of good graphs.

Export EPS – A participant who desired a format supported by LaTeX, suggested support for exporting plots in EPS image format.

Min/Max axis values – Defining min/max axis values is a cumbersome procedure when multiple Y-axes are involved. The current interface for this task involves a dropdown list, for each variable on the Y-axis, and two associated input fields. When changing the values for multiple variables, selecting variables from a dropdown list was seen as bothersome, and values or ratios relevant for other variables would often be forgotten meaning the user would have to go back in the list, and may sometimes forget which variable they were currently defining min/max values for.

Render static variables once – Profile plots have the ability to plot the same variable from multiple time-steps in one chart, this functionality duplicate the variables in the legend including static variables such as geometry, which does not change over time. It was suggested that when such plots are made, static variables should only be rendered once on the chart, thus reducing the amount of visible clutter in the work environment and in the exported image.

Adding timestamps – When plotting multiple time-steps in a chart, the user click on a button to define the time-step to add, this will also include a timestamp in the legend. However, when using the playback controls to select a time-step (displaying only one time-step) no timestamp is added to the legend. In situations where multiple charts, containing single time-steps, are exported there is no indication of what time-step they visualize (*i.e.* does the chart show the conditions for time-step 1 or 2?). It is possible to add a timestamp manually, by using the same button when selecting multiple time-steps, this is, however, easy to forget and it was suggested to change the behavior so a timestamp is always added.

Adding more variables after simulation – some participants desired the ability to add more variables to the plot after completing simulation, *i.e.* variables not defined as output before simulation, as it was common to forget to define the needed variables sometimes or not knowing which variables were necessary before performing a simulation. To make this possible it would be necessary to perform a simulation using all required variables, depending on case definitions, which would make manually defining output variables unnecessary, but would likely increase the time required to complete simulations.

Column size for description – By default the description column, in the variable selector, is located as the last column and given less space resulting in truncated text, while other columns have unused space. This was perceived as the description being hidden and requiring novice users to fix the size of the column manually to view the complete description.

P&ID – The suggestions regarding P&ID are relevant only for the 2015 version of OLGA. It was suggested using color-coding on the different model components to be able to visibly identify different types of components, and make working with the diagrams more colorful, as the new black and white style could become boring to work with over time, especially during continuous use.

The suggestion for colorization in models was suggested, as this had previously been used in some projects to make it easier to identify different parts of a system. This functionality was present in earlier versions of OLGA and it was uncertain if it still existed in the 2015 version. It was later revealed that the functionality is still present, the steps required to colorize could, however, be improved.

10.2 Focus Group

The focus group was conducted in a larger meeting room, with the participants seated around a table. For the session, a topic was defined in advance and the participants had the topic available in a written form. However, the group interview was conducted in a free form, where participants were allowed to bring up topics they considered important. Furthermore, paper and pens were provided with the intention of enabling the participants to make their own notes, draw example drafts or for other use during the interview.

In total, five users participated in the focus group, with similar ranges of experience as the individual interviews. The group consisted of three consultants and two testers.

Location	Type	Issue
Case	Anticipation	User must specify what they want before performing simulation
	Efficiency	Building a model is time consuming
Plotting	Effort	Difficult to modify the looks of plots
GUI	Visibility	Easy to lose track of which case is active
General	Efficiency	OLGA is bound to memory intensive workflows
Var. selector	Visibility	Description column is “hidden”
		Multiple columns contains unused space
	Affordance	Column header misinterpreted as a button to toggle select state
Descriptions	Documentation	Difficult to see the “English” behind the science
Output def.	Visibility	All variables, including irrelevant ones, are listed
	Efficiency	Locating specific variables can be difficult
Output def. Var. selector		Search feature, searches in all columns
Model browser	Consistency	Property descriptions varies in type and content
Model	Affordance	Components with separate dialogs do not indicate if a dialog is available
		Separate dialogs for components can be confused with dialogs to specify property values
Static plots	Bug	Results selected with value tracking not cleared when dialog closed
Diagram view	Control	[2015] Changing “right angle” to “straight line” mode, or reverse, do not convert flow-paths to angled or straight
		[2015] it is not possible to manually alter the routing of the angled flow-paths
Model		[2015] The size of node components, may be too large for use in a report

Table 10-5: list of findings from focus group

Additionally, during the focus group session, some issues previously discussed (10.1.1.3 Plotting) were repeated. The repeated issues were, *graph lines turning green, legend not displaying all plotted variables, space used by extra Y-axes* and the *overwhelming configuration window*. Another discussed aspect, is that OLGA provides an overwhelming amount of options, which may overwhelm new users.

Modifying plots – What makes a plot look good is highly subjective and functionality to perform modifications should be available. In OLGA functionality to modify plots are available in the configuration window, the functionality, however, provided is difficult to locate due to the overwhelming nature of the configuration window.

Specifying desired output – OLGA is used to discover something new and is often used when there are many unknown aspects. In such situations the demand on the user to predicting what they need is likely to result in unproductive use of time, increased risk of forgetting to specify what they need or simply not knowing that a specific variable may be required to make sense of the behavior of another variable.

Time-consuming modeling – When building a model in OLGA, there are many properties associated with a component, these properties are mostly grouped under the component in the model browser. However, some properties require other properties located somewhere else to be set. The procedure of building a model includes a lot of fiddling with properties.

Overwhelming options – As a tool OLGA must provide enough functionality and flexibility to solve varying types of tasks. The amount of functionality and options may serve as a barrier, making it more difficult for new users to learn how to use the tool.

"Hidden" description – By default, the description field in the variable selector is located on the right edge of the list, with large parts of the description truncated. Some of the other columns have a larger size than they will use, resulting in **columns with unused space**. For more experienced users, this is a non-issue as they are familiar with the keywords, for novice users it is more of an issue as, for example, the meaning behind QM or PT can easily be forgotten.

Sorting button – When working in the variable selector, one of the visible columns is for sorting the variables according to the selected/deselected state. During the session, the column was briefly confused, as a button to toggle the selected state of all variables in the list.

The English behind the science – Domain specific and some terminology specific to the OLGA is used. More experienced users may be familiar with the terminology and recognize the meanings, which may be unfamiliar or appear alien to novice users. While OLGA does provide documentation and descriptions, it is not always easy to understand and sometimes difficult to locate or connect the documentation with what is apparent in the interface. As an example of "difficulty to connect", it took the evaluator quite some time to connect the "multiple plots" button with the documentation for "interactive trend and profile plots".

Memory intensive workflows – Most of the data used and created by OLGA is stored in ASCII format, which require more disk space to store, and memory to load. In most situations, the cases created by the users are huge, with many variables defined in output, resulting in files containing a large quantity of data. When loading these files the available memory (RAM) is filled up with the content of the files, causing system slowdown and risk of crashing applications.

All output variables listed – When defining what variables to include in the simulation output, the interface presents all variables in a list. The list includes variables that are not relevant to the current case (model, components and options) and will not add anything if added. This results in an overwhelming list where it is difficult to locate the desired variables, more so for novice users, and difficult to locate relevant variables when uncertain of which variables to add in the output. A possibility is to use the search feature included in the output list and variable selector, assuming the user know what they are looking for.

Search performed on all columns – When using the search feature to locate a specific variable, the search term is used to search in all columns, meaning searching for the keyword PT will display all variables with that term, regardless of where it is located. Nothing will be filtered out if, for example, the case name or file path contain the search term. Furthermore, it is not possible to specify where to perform the search (*i.e.* which column).

Losing track of active case – When working on a case, the currently active case will always be visible in the case tab bar. However, most of the interaction is located in the diagram view and model browser, and switching between cases can become an automatic sub-conscious action requiring little attention. With the focus mainly on the diagram and model browser, and little conscious attention when switching between many cases, it is easy to lose track of which case is currently being modified. The result may be that the user performs many modifications to a case and then later discover that they are working on the wrong case, at which point they will need to recall what changes they made.

Inconsistent type and content of descriptions – When reading the description for various property keywords, the type and content of descriptions vary. Some descriptions are brief, while others are highly detailed resulting in some descriptions having a lot of unused space, while others are truncated with no indication that the description is truncated. Some descriptions also explain what type of values the property accepts, and some explain conditions for using the property correctly. The fact that the type of descriptions vary is understandable, considering they cover different types of properties. The detailed descriptions suffer from truncating text without indicating that the text continues, while the brief descriptions may be overly brief and simply reused in the documentation.

Separate settings dialogs – Some components, such as flow-paths and centrifugal pumps, provide a separate dialog for some settings. There are three ways to open these dialogs: double click the component, a properties button on the toolbar, and a button located in the model browser's property editor. There is no visual indication that double clicking any component will open a separate dialog, for most components the behavior will be the same as a single click. The properties button on the toolbar is always active when a component is selected, for a node it will simply set the focus on the corresponding location in the property editor, for a flow-path it will launch a separate tool for geometry. The button located in the property editor is disabled for all components, except those with a separate dialog; this specific button is the only indication that a component has a separate dialog. However, it is easy to overlook the button, due to the location and size, which may result in users not knowing the dialogs, exists. The separate dialogs can also be confused as alternative dialogs to specify properties.

Tracked values not cleared – when extracting data from a specific point value tracking is used, the feature open small dialogs with the data at the specific point. When these dialogs are closed it is expected that the values are cleared, *i.e.* not included when copying the data from the other dialogs. However, when closing some of the dialogs and copying the data from the other dialogs, the data from the closed dialogs are included in the copied data.

Furthermore, some discussions focused on categorizing the various output variables, due to some participants sometimes struggling to locate the needed variables in the output definition, and it was assumed that categorizing variables would make it easier to locate variables. During the discussion, it was revealed that output variables were already grouped according to a main-groups and sub-groups, it was further revealed that the main-groups were of little groups, due to being too broad (*i.e.* the group *Basic* was the largest group), and that sub-groups should be used for sorting. This does indicate that it might be beneficial to revise the current groupings and make the groupings more visible, *i.e.* a main-group consists of variables for a certain purpose, which can be derived from the group label.

The 2015 version of OLGA included some changes; the major visible change was the switch to P&ID. With the P&ID styled components, the node components may appear too large and appear to waste a lot of space when using the model in a report. The behavior of flow-paths has also changed, by default, the mode "right angles" is enabled and the alternative is straight lines (Figure 10-5). In older versions, when right angles or straight lines are enabled the flow-paths are automatically adjusted. In the 2015 version the flow-paths are not automatically adjusted and to switch between the appearance of the flow-paths the user must first delete the unwanted (straight or angled) flow-path and insert a new flow-path in the desired mode, this require unnecessary effort and is not always an option when all the necessary adjustments to the flow-path has been made. Furthermore, in previous versions, it was possible to click and drag the angle and manipulate how the angled flow-path would route in the diagram, given the example in Figure 10-5, the user could manually decide if the flow-path out of the inlet node would exit horizontally or vertically. However, it must be noted that, according to the developers, the routing logic for flow-paths was updated to avoid flow-paths and components from crossing over each other in the diagram, Figure 10-6 illustrates how components and flow-paths could cross over each other in the diagram.

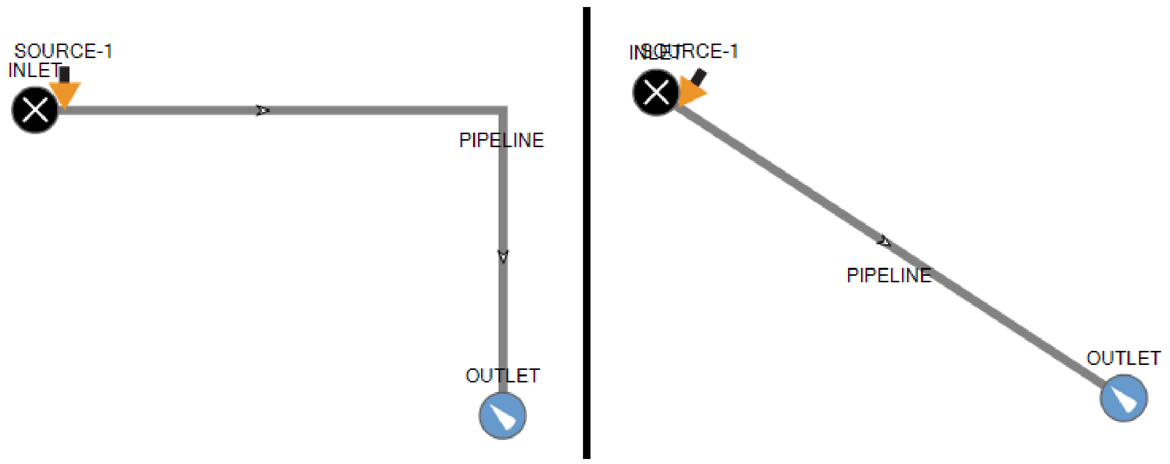


Figure 10-5: illustration of angled (left) and straight (right) flow-paths

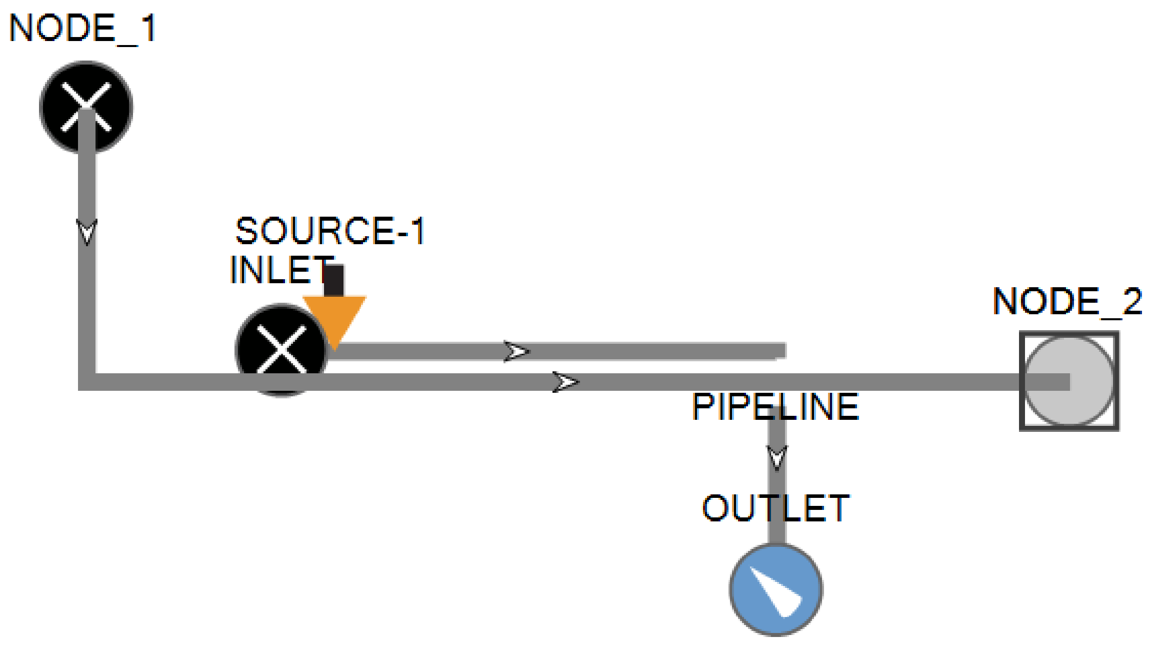


Figure 10-6: example of crossover of flow-paths and components in models

10.2.1 Requests and Suggestions

In this section, requests and suggestions made by the participants are discussed.

It was suggested to remove the plotting functionality of OLGA, and provide an interface to acquire the simulation data from third party tools. The reasoning was that, in the participant's mind OLGA was a tool to make the calculations and should focus on that task, while making it easy to fetch the data, and make automatic creation of plots possible in other tools. Other users may not agree with removing the plotting functionality from OLGA, however, the suggestion of an interface could make it easier and more efficient to extract the desired data and process in the users' desired tools for such purposes.

In order to better understand behavior in production systems, and understand how changes propagate, the ability to plot along a network and not only a branch was suggested. Due to the variance of the layout of production systems, this is a suggestion that could deserve its own research study in order to determine: how to determine the confines and possibly alternative visualization or methods to present the data in an understandable and usable way.

A suggestion aimed at reducing errors, where the user has forgotten or was unaware that one or more output variables were needed. The suggestion was to make OLGA run simulations using all relevant variables without the user needing or having the ability to specify output variables. Furthermore, as this change would result in more data being stored in the result files, it was suggested to change the storage format from ASCII to binary due to the second providing benefits in reduced storage overhead, and increased writing and reading speed.

Another suggestion for variables, when using the plotting functionality, was the ability to add new variables "on the fly". The desired functionality would allow the user to perform simple calculations on existing variables and create new variables with the calculated result, without specifying this before running a simulation or using third party software. An example of the desired functionality: take the pressure from one point and the pressure from another point and perform, for example, a subtraction that would result in a new variable with the difference between the points.

It was also desired to be able to retrieve the max/min values of variables, without needing to define this before running a simulation, manually attempt to locate the values, or using third party software. Such functionality is already available in OLGA, hidden away in the configuration window and difficult to locate. The existing functionality also provides mean, median and average values.

OLGA mainly uses trend and profile plots to present the simulation results, which is also available as tabular data. However, the data table is only available in the configuration window, and provides no ability to copy data from rows, columns or cells. It was suggested to provide a tab with the simulation results in a spreadsheet, it was stressed that OLGA should not attempt to recreate functionality in software such as MS Excel. However, presenting the data in a spreadsheet was viewed as a method where it could be easier to extract specific data, as an alternative to value tracking, and possibly add some more functionality (specifics were not mentioned).

To make it easier to see specific plotted variables it was suggested to include functionality to highlight a specific variable. As described, the functionality would make it possible to either click on an entry in the legend, which would visually highlight the associated graph line in the chart. The functionality could also provide the reverse, *i.e.* user click on a graph line and the associated entry in the legend is visually highlighted.

When selecting variables in the variable selector, it is possible to specify which unit to use. When specifying units for multiple variables, the procedure must be performed individually for each variable, which is inconvenient and take more time. It was suggested to provide an ability to select multiple variables, with the same unit types, and change the unit for all selected variables. Thus, the procedure for changing the units for ten variables would consist of *highlighting* ten variables and change the unit on one of the highlighted variables, which would automatically use the selected unit for the remaining variables.

Furthermore, for static plots it was suggested to implement functionality that would automatically refresh the plot, without user interaction, when a simulation completes. In the currently available versions, the user must manually click a refresh button to update the plot with the new results. However, such a feature should be possible to toggle, as in some situations it might be desired to keep the chart for previous simulation runs.

OLGA comes bundled with third party modules such as Rocx and FEMTherm, OLGA is able to use the data from these modules and in the case of FEMTherm able to plot the data. However, OLGA was not able to plot data from modules such as Rocx. Thus, a suggestion was to make OLGA able to plot data from bundled modules, without requiring installing extra software. The reasoning was that OLGA is already able to use the data, and thus should be able to plot the data, as well as reducing the need to install and learn how to use other tools to visualize some data.

In older versions of OLGA, it was possible to change the node type by editing a property, in the 2015 version this was no longer possible requiring the user to manually delete the node and add the desired node type. It was unclear if the change was intentional or a result of the changes in the 2015 version, it was however viewed as inconvenient and the previous ability to easily change node type was desired.

To make it easier for new users to start using, and becoming familiarized in the domain, it was suggested to develop a *light* version of OLGA. The OLGA Light version would provide, according to the suggestion, a simpler interface and provide presets and other modifications to reduce complexity. The intended target group was new users and users who may not need all the functionality of OLGA.

The search functionality provided in the output definition, which is similar to the search functionality in the variable selector, was also discussed. The search feature in both dialogs performs searches in all columns, which can cause issues when searching for a keyword that may be present in other columns for all variables. It was suggested to add the ability to specify which column to perform the search, which would make it possible to filter according to specific criteria, such as searching for a keyword QM in the variable name, which would ignore all other columns that also might include a word with the search term.

Among the most important factors, for users, is stability and efficiency. Both have improved in newer versions of OLGA. A specific request was to improve the loading times when starting OLGA as well as when loading the various files, loading times (and stability) may be more relevant for the internals of OLGA as well as the storage formats used, it is however considered relevant to mention as users *experience* the issues related to these factors.

In relation with the right angle functionality in the 2015 version it was desired to be able to manually correct angled flow-paths, as well as making flow-paths automatically convert to right angle or straight line, depending on which mode is active.

10.2.2 Other Topics Covered

The participants also provided insights and other suggestions relevant to mention.

When questioned if interactive plots were commonly used among the participants, it turned out that the participants frequently used static plots, and only touched interactive plots when performing support. This was expected, as interactive simulations and plots appear to be more suited for observing the conditions in live production systems.

It was suggested to perform a case study with comparison between OLGA and LEDAFlow, specifically on the amount of time required to build a model. This would make it possible to discover where the users spent time, where problematic procedures exists, as well as see what aspects of the interfaces works well. Some of the discussion included adding some functionality from another Schlumberger product (PipeSIM), such as using a real map as backdrop for the model. It was agreed that using a map backdrop had no necessary purpose, and would not improve working with the models, due to the abstractions used in modeling. However, despite this it was mentioned that, though a map backdrop served no purpose, it might be nice to show in presentations to give a better feel of how the production system looks like. Furthermore, it was mentioned that competitors focused on providing simpler interfaces and provides a lot more defaults, which serves to make it easier to work with the interface, as well as reduce the time needed to build models since the more common properties are defined. In relation to the discussion surrounding competitive software, it was mentioned that performing comparative studies with competitive systems, including systems that work with complex models and property definitions not related to the domain, doing this would make it possible to see what aspects of OLGA works well, and aspects where other systems performs better. Furthermore, comparing with non-domain systems also provides the opportunity to see alternative solutions that are distinctly different from the existing solutions in the domain. Comparing with other systems has the added possibility to encourage trying something new, rather than sticking to the same old solutions.

The participants receives specifications and details from customers, which takes a long time to prepare, users mainly interact with OLGA using a combination of mouse and keyboard, when preparing input for OLGA some use external tools such as MS Excel, which can be used to prepare inputs for tools in OLGA which accepts/use tabular data.

10.3 Evaluator's Observations, Experiences and Discoveries

During the interviews, and familiarization with the system a sizeable amount of possible improvements were discovered.

Missing licenses – OLGA requires a license, as do some of the external tools, which require an additional license. If using a network license for OLGA, it is possible to lose the connection with the license server. If the connection is lost, and it is attempted to run an interactive simulation, nothing will happen, no feedback or warnings, nothing appears in the output tab, the interface will freeze up for a bit as it usually do and then unfreeze without doing anything. There are some alternatives for how the OLGA GUI could solve this situation:

1. When launching OLGA, it could check for a license, if license exists launch time it is assumed to exist for the entirety of that session, regardless if the connection fails.
2. When attempting an interactive simulation, OLGA checks for a license, if not found a warning message is displayed indicating what is wrong, this lets the user know that the license could not be found and can perform the necessary steps to rectify the situation. The message could also include a button for retrying.

With external tools, that requires separate licenses the issue is different. When launching an external tool, the user must wait for the tool to load and check for a license, when the check fails it means the user just spent a certain amount of time to find out if they could use the tool. To avoid such situations, the tools-page in OLGA could benefit with indicators for the license state, the indicators could be used to communicate if the tool require a license or not, and if the necessary licenses are present or not.

Furthermore, some of the sample cases includes aspects that require separate licenses, the sample descriptions do indicate, textually, what licenses are required to fully use the sample. However, it is not always apparent for the user, which licenses they currently have available, and users could benefit from an indicator communicating the presence of licenses. In the 7.3.4 version, users do not find out if they are missing a license before attempting to perform a simulation. Additionally the behavior of interactive and batch simulations are inconsistent, a batch simulation notifies the user of the issue and refuses to run. While an interactive simulation notifies the user of the issue and informs that the parts relying on the missing license will be ignored, and performs the simulation.

Cases and Projects – When creating a Case OLGA will automatically create a temporary project, when closing the single Case, with the close project button, or when closing OLGA, the user will always be prompted to save the project. This prompt may become tedious when frequently creating standalone cases, a possible improvement could be providing configuration options for the behavior, *i.e.* if a prompt for saving temporary projects should be displayed. Another possible issue with Cases and projects is that Cases do not know if they belong to a project, OLGA will not check if a Case belongs to a previous project, which, with lack of attention, can lead to one Case being associated with multiple projects. Cases associated with multiple projects can lead to situations where the case is modified to suit one project, and later be modified to suit another project, which upon simulation can lead to simulating the wrong aspects (if the "wrong" project is loaded).

File menu – Some entries in the file menu, requires that mouse clicks are performed within a specific area (detailed in 8.1.2 Pages). This behavior is most notable in the create buttons located on the new-page. However, it is also present in almost all entries in the various pages, though with smaller areas where a mouse click will not initiate the action.

Lack of control – When empty cases are created, the user have no control over the name or the storage location. By default, empty cases are named "case", or if the file name already exists in the directory, OLGA will append an integer to the name; the result is that a directory may contain a bunch of files with non-descriptive names. The user has some control over the storage location when a project, and the directory can be defined, is created before the Case. The way OLGA behaves is that, when there is no active Case or project, it will use a pre-defined storage location. When a project is opened or created, OLGA will use the project's directory as storage location, and if one or more Cases are open OLGA will use the directory of the first opened case as that is the directory belonging to the temporary project. Additionally, when creating an empty case and later using the save project button, the user cannot define the project name or the storage location, the location and name of the Case will be used. A workaround available to achieve more control, is to use the *save as* features, however, for Cases this will create a new copy of the Case resulting in duplicate files with different filenames. For projects already saved with *save project* it will create a duplicate project file, referencing the same Case files as the original project. For temporary projects it will provide the expected functionality, *i.e.* provide control over project name and storage location, case files will not be copied.

Fluid properties table – A PVT table with fluid properties is required to perform simulations, as mentioned earlier, OLGA includes an external tool to assist with creating such a table, and sample cases provides a pre-defined table of fluid properties. However, if neither of those options can be used, there is little support in the documentation, for novice users, in how to create such a table manually.

Defining Materials and walls – Models will most likely always need some defined materials, and wall constructions, and some might require special entries, which is located in the Model Browser, under the library branch in the tree-view. However, for new users, without access to training materials, the procedure to define these aspects of the Case is unintuitive. The interface does not indicate where or how to add such definitions, attempts to check the manual were initially of no help and in the reviewers situation it took around six months and an unofficial guide to discover how materials and walls could be defined. In hindsight, adding materials and wall definitions is not difficult, simply right-click on the library branch and select the relevant entry in the *add* sub-menu.

Property fields – The property fields can be frustrating, as previously mentioned the sorting of properties can be defined (8.2.5.2 Property Editor), by default the sorting used is *Used Keys* and will include a *not used* group. The behavior of this sorting result in property entries jumping around, specifying a value in one property may result in another property not being needed, and the unneeded property is then moved to the *not used* group, resulting in the property list changing. This behavior can be disorienting, and in some situations, it is possible that the property being edited change location, this can be improved by using the *Complete* sorting, which provides almost the same sorting, without the "*not used*" group.

Furthermore, when invalid data is entered in property fields the property editor will silently change the value, without notifying the user, *i.e.* entering text in a field requiring numeric data will silently change the text to 0 (zero). If the user is inattentive, this could result in simulations with incorrect results, an improvement could be to *clear* the property field and display a balloon message with an explanation as to why the field was cleared. Another behavior is that property values are automatically converted when changing units on properties, this could be made configurable where the user could decide if the behavior when changing unit should be *converting the value* or *keep the same value* (*i.e.* convert 10mm to 0.01m or change 10mm to 10m).

Additionally, it is unclear if the colors used to convey the state of properties (Table 8-2) matches the documentation in behavior, which creates situations where it is unclear if a property will be used or not, or if entries in the *not used* group may be optional. According to contacts at Schlumberger, properties in the *not used* group will not be used during simulation. However, during use it has been observed that some properties will be relocated from the *not used* group to the *used* group when the property value is set. Modifying the behavior may improve the situation, such as required properties being assigned a color which is always used, regardless if the property has a specified value or not. Unneeded properties could be grey and refuse input, to avoid confusion if they are used or not during simulation. If a property is suddenly needed and optional the field could be enabled and use a black color, this could be used in combination with symbols which can be used to convey the same information, resulting OLGA not relying on only color to convey information.

Inconsistent terminology used to refer to interactive plots – How interactive plots are labeled and referred is not consistent. In the toolbar, the tooltip associated with the button is labeled “Multiple plots”, the tab for the interactive plot is labeled “Plot” and the default heading for the plot is “Custom plot”, both multiple plots and custom plot also used in some parts of the manual. In the manual interactive plots are referred to as “interactive trend and profile plots” and “interactive plots” in headings and the index. Most of the alternative phrasings are by definition correct; users are able to create multiple charts in one plot, users have the ability to customize the layout and change the type of plot from a line graph to, for example, a pie chart, and to plot data one must run an interactive simulation. However, for a new user who would see “multiple plots” first and then the title “custom plot” and if nothing is plotted, when a batch simulation was performed, the terminology will not match the corresponding heading in the documentation. This inconsistency requires that the user associate the words “multiple” or “custom” to “interactive”. Furthermore, searching either multiple plots or custom plot will not provide any information on how to plot the data.

Furthermore, the documentation do not clearly state how to plot data, it guides through adding a plot and details about possibilities and limitations. However, the documentation do not clearly state that, for data to be plotted an interactive simulation must be run, there is no mention of interactive simulation in the section for interactive trend and profile plots.

Interactive plots – When data has been plotted and the variable selector is opened, the plotted data will be cleared, even if no changes are made to the selection or the cancel button is pressed. Furthermore, the context menu includes two toggle entries for “Show border”, which is only relevant when having multiple charts in a grid, and “Show variable selector”, which have no observable effect.

Both interactive and static plots provide the ability to specify a theme for the plot, which can be used to make the plot look better or be easier to read. However, it is not possible to specify a default plot theme for both interactive and static plots, this is not a requirement but would improve efficiency if a user has found, or created, a theme which suits their needs (aesthetic, legibility, visibility, etc.). Additionally, a bug in the context menu, for interactive plots, was discovered. Located in the “copy” sub-menu is an entry for “All images” in some instances there will be two “All images” entries, where one of the entries do not copy anything.

Interactive plots provides functionality to change the type of chart, some of the available types are pie chart, bar chart and an analog clock. How useful the available charts are, in terms of flow assurance, was not evaluated as none of the participants used interactive plots. However, some of the available types break the plotting functionality, and plots, with the result that a user must manually recreate the plot.

Furthermore, when a plot is removed there is no method available to undo the removal. The result is that, if a plot is accidentally removed from a grid or a plot tab is closed, the charts are lost and the action cannot be reversed, to restore the plot it must be manually recreated. For interactive plots this means that, the plot must be created and an interactive simulation must be rerun. For static plots this means that the plot and any customizations must be recreated, it is not necessary to rerun a simulation, as the data is already stored in files. The severity of this for interactive plots is unknown, though assumed low, as the module is suited to plot live data and new plots can be added while the interactive simulation is running. This means that only the data between the removal and re-adding of a plot, is not plotted but not lost as it is most likely stored in files.

Documentation – In some situations, the description for a property may be too brief and more information about the property may be needed. However, for some properties the documentation only repeats the description, for example, a property may be labeled "HYKPLIST" with the description "Array element of HYKPLIST" which is repeated in the documentation, no terminology list giving the long-form is provided, leaving it up to the user to decode the meaning. This may be more relevant for novice users and users who are exposed to new properties, as they may not be familiar with the terminology, while the meaning of the various keywords may be obvious to more experienced users. Furthermore, the description area in the property editor may not fit the complete description, and do not indicate that more text exists. Additionally, the documentation provides some concrete steps to complete certain tasks, such as getting started videos illustrating how to perform common tasks. However, the documentation is more focused on providing technical details and background for the various topics.

10.4 Universal Design

Universal design did not have a central role during this project, mainly due to initially starting as a usability project. Furthermore, discussions revealed a desire to make the system more usable for the current users, and there was no obvious focus group at the time. However, some usability issues, which may function as barriers making them universal design issues, were discovered during the familiarization with the system, and interviews. Some of these issues have been described earlier, but is briefly repeated here.

The GUI and Plots had issues with color contrast, where graph lines had too low contrast between other lines and the in some circumstances the background. Some color combinations could also be problematic for users with color vision deficiencies, as well as situations where the chart is limited to presentation in black and white. Static plots provide functionality, a black and white mode that partially improves the situation (see Figure 10-7). The mode use black and grey color combinations with better contrast ratio as well as variations in the line patterns. A drawback with the black and white mode is that the colors are not used on the Y-axes, resulting in axis-labels that do not provide meaningful information. Furthermore, some line patterns can be too similar to the dotted gridlines in the chart, and some graph lines can look similar to other graph lines.

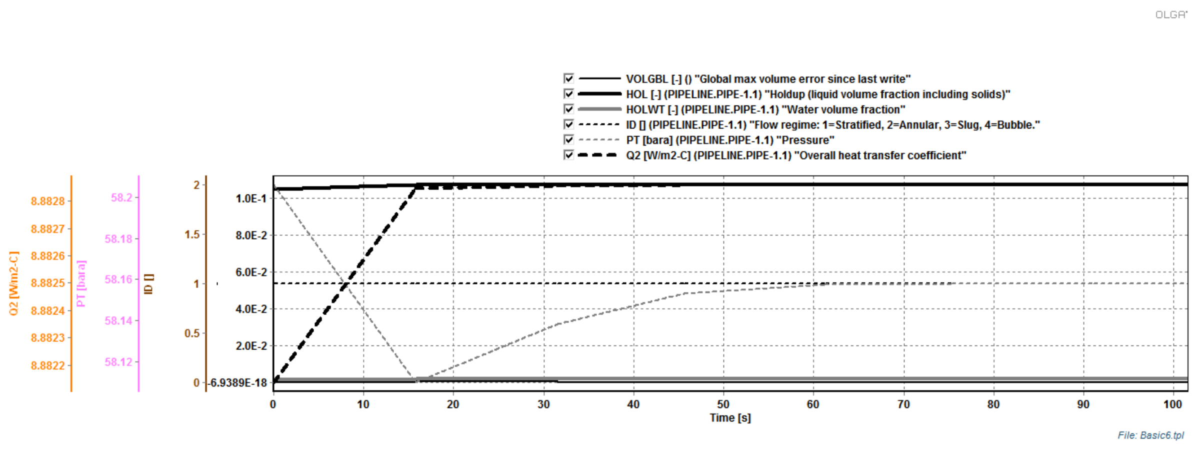


Figure 10-7: Sample of black and white rendered plot

Contrast issues in the GUI are most apparent on the help-page, where disabled entries, such as “send to support”, use a light grey text color, the same light colors are used when the “getting started” entry is focused. These light text colors may be difficult to see even for users without vision deficiencies.

The terminology used for labels in plots is highly OLGA specific, resulting in reduced access to the plots for users *of the plots* who may not be familiar with OLGA or the terminology used. The size of exported plots makes it difficult to see details the exported plot when used in reports, as seen in Figure 10-7. Furthermore, the legend label can be difficult to read due to the text pattern and font size. A majority of the entries in the various pages in the file menu requires that mouse clicks occur within a specific area on the entry, if a click occurs on the entry but outside the click area, no action is performed.

Furthermore, the OLGA GUI has some issues with associating tools in separate windows with specific Case tabs. This can be problematic in situations where users have many cases open that are frequently switched between, or when external distractions take away the focus. Another problematic aspect is specifying output variables before running a simulation. When the dialog is opened all variables, including irrelevant ones are listed, which increases the requirements on the users in remembering/knowing which variables are relevant for the current model and case definition, as well as anticipating which variables may be needed after a simulation has completed.

Other aspects not supported are keyboard navigation, ease of access platform features such as high contrast themes, font settings and external AT such as screen readers. The MS Windows platform provides *access keys*, which are functionality similar to keyboard shortcuts; the difference is that access keys are used to accessing and navigating the UI, while keyboard shortcuts are means to perform common actions. Supporting the platform's ease of access functionality has some benefits, such as making it possible for users, who needs the ease of access functionality, to use OLGA, as well as being consistent with the platform. Furthermore, supporting the functionality may also improve the overall user experience of the platform.

11 Suggestions

Some improvements could be made to the documentation and interface of OLGA.

11.1 Documentation

Provide instruction for how, when, why and dependencies of use, the current documentation already provides much of this information, however it may be difficult, for a novice user, to glean the *how to use* for elements. For example, for interactive plots the documentation could provide specific steps to add the plot, selecting variables, and performing an interactive simulation to plot data.

Provide an overview of keyboard shortcuts, some common actions have associated shortcuts that may or may not be advertised in tooltips or documentation. Providing an overview makes it possible for users to identify and use relevant shortcuts.

Provide links to further information or to relevant topics, some topics are split into multiple sections in the documentation providing details, with the current manual the topics are only easy to reach when using the search function, with the correct keywords. By providing links, it would be easier for the user to locate the correct information, for example if a section provides a brief introduction/overview of a case element with multiple topics it could for each topic, link to a more detailed description.

Avoid reusing the property descriptions used in the Model Browser, and give more detailed information. Links to more detailed descriptions could be used when sections provides overviews of relevant elements, *i.e.* a property may be included in the overview available in multiple sections, then each section could provide a link the properties main page where the detailed information is available. Each property description could additionally include a link to the location in the manual where detailed information is available.

11.2 Main Interface

OLGA could provide *access keys* enabling access to UI elements, and provide more keyboard shortcuts for common actions in OLGA. The file menu could be made reachable with the keyboard as well as providing tab navigation in the file menu without requiring mouse input.

Include a close button on case tabs. With the intended action to close a case tab, not delete or remove from a project, there would be need to be some modifications to how OLGA handles cases. When the close button is clicked, it should:

1. Check for modifications to the case since the last save
2. If modifications were found prompt user to save
3. Close the case, but not remove it from a project

For consistency, the behavior should be the same for multiple open cases not belonging to a project and cases belonging to an active project. This would require some changes to how projects are treated in OLGA. For example, a project manager dialog could be added to the UI listing all the cases associated with the project, this could simply be a list of entries with some checkbox options such as:

- Include in project batch – case file is added in project batch simulations even if the case is not open in OLGA.
- Open on load – toggles if the case file is automatically opened when the project is loaded.

Additionally, a context menu could for each case include an entry for the two options listed above, as well as entries for *Close case* and *Remove case* where the first would just temporarily close the case while remove case would provide the ability to remove the case from the project and optionally all files belonging to the case.

11.2.1 Model Browser

Some modifications that could be made include preventing properties from moving when their state is changed. Keep indicators for required, optional and not used states when values are set / unset, *i.e.* if a value is set on a required property, it should still keep the indicator for required properties.

Properties for a component that require the presence of another specific property should, either be disabled, preferably with a tooltip indicating how to enable the property, or not visible at all depending on the presence of the dependent component.

Avoid silently modifying property values, on invalid input the input field should restrict the type of input to the required data type (for example, accept only numeric input). Alternatively, the value in the input field can be cleared and a local message, explaining why, can be displayed. Furthermore, the behavior of the property editor could be modified, to set focus automatically on the input field when a property label is tabbed to or clicked on.

Property values requiring tabular data could, display a dialog with a table for data input when the input field is activated. Similar behavior can be used on properties accepting lists. By displaying a dialog when the input field is activated it may reduce some confusing situations such as if a list is comma separated, the amount of columns and rows in the table and how to format the table data. A dialog would also provide the additional ability of the UI to instruct if the input requires modifications on other properties.

The ability to toggle the floating state of the model browser window could be included, where the user could decide if the window should be floating or static. The static mode would snap the window to one of the edges and resize the diagram view, which would result in the model never being covered by the model browser.

11.2.2 Output Variables

The requirement to specify what variables to modify could be changed, as well as the storage format. The proposals below assume that the storage format is changed to binary:

- **Simulate all relevant variables** – For a given case, simulations are performed on all relevant variables. This may require longer simulation times, but would reduce user errors that could potentially require a simulation to be run again after completing the first run.
- **Opt-out variables** – Similar to the alternative above, but provides users with the ability to deselect variables. This may provide the capability to improve the simulation time as well as introduces risk of user errors, *i.e.* deselecting a needed variable.

11.2.3 Model

Components with specialized dialogs could provide a visible indicator, on the component in the diagram, that it provides a dialog. This could be a button, which is either always visible in OLGA or visible when the component is focused or hovered.

The functionality to specify colors for parts of the models could be modified to make it more efficient, *i.e.* rather than requiring the user to first specify a color, name the color, then apply the color on the model it could be reduced to applying the color when the color is specified. Additional functionality to name a color could be made an optional step.

11.3 Plotting

The most basic fixes that should be made with the plotting functionality, is to fix the issues causing the legend to be incomplete and graph lines defaulting to a green color when plotting over ten variables. Furthermore, editing labels in the legend and axes can be made more efficient with one of the following alternatives:

- Make the label UI element editable – clicking on the label allows user to edit the label directly in the chart without a separate dialog.
- Provide an edit button – clicking the edit button could turn the label into a local text field or launch a separate dialog. The button itself would only be visible in the UI and not in exports.
- Provide an *edit labels* entry in the context menu – the dialog could provide a two-column list where one column contains the default label and the second column accepts input for the custom label, if the custom label is empty then the default value is used.
- Provide an ability to highlight one or more legend entries, which would make the associated graph line more visible, by making the line(s) thicker for example. The reverse functionality could also be implemented, *i.e.* selecting graph lines result in the increased visibility of the associated legend entry.
- Provide color and pattern selectors in the toolbar and alternatively in the legend. The selectors in the toolbar would apply the change to the currently focused graph line, if the previous proposed functionality is implemented. In the existing UI the legend includes a visual identifier of color and line pattern, this indicator could be made

interactive where clicking on the indicator a dialog with functionality to define color and line patterns.

- Develop a custom plot configuration window for OLGA, the current configuration provides unnecessary functionality and is overwhelming. The custom configuration window could be split in specific tasks, such as defining font settings, chart types, export settings, specifying themes, and more. It should not be necessary to port all the available configuration settings, providing good defaults and some configurability to the presentation. Themes could be used to provide the more detailed and flexible settings of the plotting functionality. The idea is that OLGA provides a good default plot with some configurability and functionality, and the more detailed aspects are left to themes, which users may develop.
- Most of the suggestions above illustrate that the plot presentation in OLGA is an interactive interface where the visible elements in the plots can be used to perform an action. This requires a separation of the UI plot and exported plot. The UI plot could use a layout optimal for editing elements directly in the interface without separate dialogs, while the exported plot could provide different layouts for specific use cases, and remove visual elements that are only relevant in the UI (*e.g.* checkboxes in the legend). Furthermore, the legend in the UI could be a floating window that can be moved around on one or more monitors, when the plot is exported, it could then automatically be placed in any location depending on the export layout. The ability to create custom export layouts could be provided as an extra tool in OLGA.
- When working with cases with multiple simulation result files, OLGA could detect the amount and allow the user to select which files to load before loading the files. If a case includes fifty result files then the interface could either provide an opt-in or opt-out dialog, if opt-in the user must specify which files to load, if opt-out the user must specify which files not to load.

11.4 Additional Functionality

Some additional functionality that may be beneficial for OLGA is briefly discussed in this section.

OLGA could provide additional functionality, allowing the users or third parties, to create custom dialogs and wizards for actions and procedures, which could then be added to context menus, automatically launch on specific conditions in the UI or be assigned to shortcuts. For example, when a case is created a custom wizard could be launched, where the user fills in required input and performs tasks such as adding material definitions so that some aspects of the case definition are completed before building the model.

Additional functionality for automation and macros could also be implemented. With automation functionality, it could be possible to define what OLGA does after completing a simulation, such as automatically create a set of plots or automatically export the data for specific conditions in a predefined format to a file or external system. With macros users could automate frequent actions and tasks, for example with the switch to P&ID in OLGA 2015 some of the feedback was that convenient functionality such as converting node types was removed, with a macro similar functionality could be re-implemented by the users.

Furthermore, simulation results could be made available in a spreadsheet, implemented as a tab. This would make it easier to copy data from specific points, without being required to locate the data manually in a plot. Additionally, adding new variables “on the fly” could be implemented. The functionality would allow users to create new variables based on computations on existing simulated variables, without needing to define the computation before simulation or using third-party tools to calculate.

12 Concept Designs

In this chapter, we introduce some concept designs that were inspired and based on information gathered during interviews and experiences with OLGA, which focus on the layout of the GUI and an alternative concept for the model browser. The designs were presented and discussed in a meeting with the contacts at Schlumberger. However, the designs were not user tested due to delays.

12.1 GUI Layout

The layout use a different approach to building and defining cases, and plotting, which is based on modes that is intended to be relevant to specific parts of the workflow. A wireframe illustrating the layout can be seen in Figure 12-1 followed by a description.

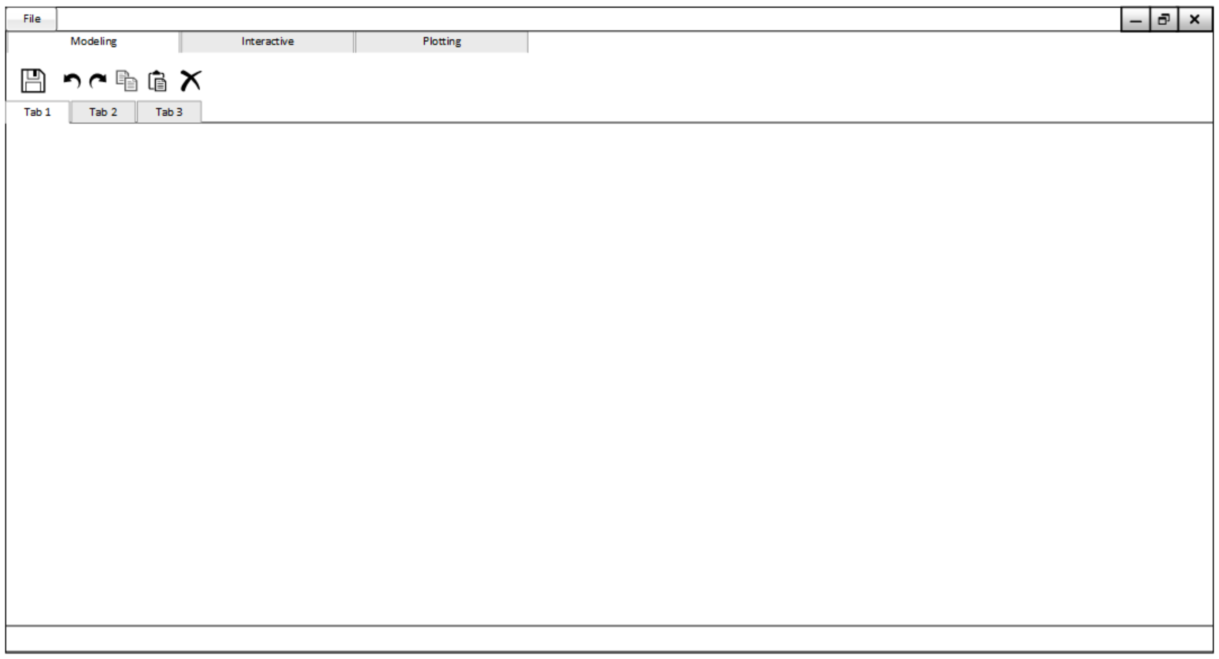


Figure 12-1: wireframe illustrating the layout

The layout consists of six main elements, which are:

- Title bar
- Mode bar
- Toolbar
- Tab bar
- Client Area
- Status Bar

The title bar is a standard element available in most window applications. However, to use the space more effectively the file menu button has been moved into title bar. The intended contents of the title bar are: the file menu button, application string and standard minimize, normalize and close buttons. Additionally, the application string is intended to include the project name.

The mode bar is intended to contain tabs for different modes. In the conceptual layout, modeling and defining cases, as well as plotting, are approached as modes, which in the wireframe is represented by the tabs: *modeling*, *interactive* and *plotting*. The naming of these modes needs more reflection and study, as the meaning of interactive and plotting both deals with plots, which is not apparent from the label interactive. Providing different modes introduces new possibilities, such as navigating directly to a mode and load a case within that mode, without loading the complete case, or adding more modes, for example Risk Management and Optimization (RMO). Another possibility is to make the visibility of the mode tabs configurable, *i.e.* if interactive is never used it can be hidden and reduce the amount of visible elements. Furthermore, the intention is that the toolbar, client area and status bar depends on the current mode so that the visible elements are relevant for the active mode. Splitting the GUI into modes also includes some challenges, such as how the case tab in modeling is associated with a plot in plotting mode.

The toolbar is intended to be dependent on the current mode and perform actions on the currently active case tab (in the tab bar).

12.1.1 Modes

Below are descriptions of the modes presented in the conceptual layout:

Modeling mode is intended to provide the UI elements and objects necessary to build and define a model. The toolbar would include buttons for actions and tools needed or useful for building the model. The tab pane would include tabs for cases, allowing for switching between different models, which requires that the toolbar belong to a tab. Likewise the content of the client area is controlled by the mode and belongs to the active case tab, in the modeling mode the client area consists of the diagram view, components browser and model browser.

Interactive mode is intended for interactive plots and assumes that interactive plots are limited to only one case at a time, *i.e.* it is not possible to run multiple concurrent interactive simulations. The toolbar would include relevant functionality for loading the case, define the desired plots and initiate an interactive simulation or step simulation. The tab pane is intended to contain tabs for plots, making it possible to define multiple plots and switching between these plots. The existing ability to create grids, with multiple charts, is intended to be kept.

Plotting mode is intended for static plots and assumes that the user may want to plot results from multiple sources. However, the tab pane would be used to separate between different plots, and not cases. The toolbar would be similar to interactive mode, as it provides functionality for loading results from cases, defining plots and initiating simulations. The mode could provide all of the available simulation types, the most relevant however, may be batch simulation, meaning that the toolbar presents the batch simulation button with a dropdown menu containing the other types. Buttons for trend, profile, fluid and 3D plots is also intended to be included in plotting mode. The plotting mode could be considered as a separate mode disconnected from specific cases such that when multiple cases are open in modeling mode, the user will be offered to specify which of the cases to use, or alternatively load a different case for plotting.

12.1.2 Navigation

Navigation in the conceptual design is intended to support pointer and keyboard navigation. Pointer navigation is implemented with actionable UI elements, while keyboard navigation is implemented with access keys, arrow keys and tab support.

When a designated keyboard key (*e.g.* alt) is pressed the interface enters a navigation mode, where all reachable elements display the corresponding access key to be used, for example with the application-modes the ‘m’-key is used for modeling mode, ‘i’-key for interactive mode and ‘p’-key for plotting mode. When the interface is in navigation mode pressing an access key will activate the corresponding application-mode and allow selecting UI elements with tab or arrow keys and activate the selection with the space-key or return-key.

12.2 Model Browser

When building and defining a case much of the interaction with OLGA occurs in the model browser. The following concept designs are intended to provide an alternative solution for the model browser. The concept does not include all possible use cases and was primarily designed with the aim of reducing "fiddling" when defining a case and avoid interaction in two UI elements (prevent selecting component in tree view and edit in property editor). Two additional goals were to reduce the required screen real estate, and avoid movement of the properties when setting property values. To achieve these goals, the design use accordions and treats components as groups with properties.

The editor is arranged in different categories, separated with vertical tabs. The categories included in the designs are *case*, *model* and *library*. The case category presented in Figure 12-2 contains the entries from the *Case Definition* branch in the existing model browser, it may include additional properties for defining changes in conditions (*e.g.* drop in pressure) and other factors influencing operations in a production system. The model category (Figure 12-3) is intended to contain components and elements that are included in the visual model. The category share similarities with the *Flow Component* branch in the existing model browser, the difference is that nodes are intended to be separate groupings not grouped under flow-path components so that it is not necessary to expand the flow-path to reach a node in the editor. Located below the list of property groupings is a pane for descriptions

The library category (Figure 12-4) is different from the other categories in that accordions are not used, but rather a list of keywords defined by the user. The entries in the list are identified by the label, which is defined by the user, and includes a link, which opens a dialog for editing the properties of the keyword. Below the list is a small toolbar, with three available actions: *add*, *edit* and *remove*. The add action opens a dialog to add a new keyword and provide a dialog to specify the type of keyword (such as material or wall) and define the properties. The edit action is only actionable when a keyword is focused, and opens the previously mentioned dialog with all the previously filled out values. The remove action is only active when a keyword is focused, and should prompt the user for confirmation when used. Furthermore, the library category could also use accordions to group the different types of keywords so that materials and walls are organized and separated according to the keyword type.

The design assumes that all relevant output variables will be simulated, which removes the need for a separate output section, this can however be easily implemented.

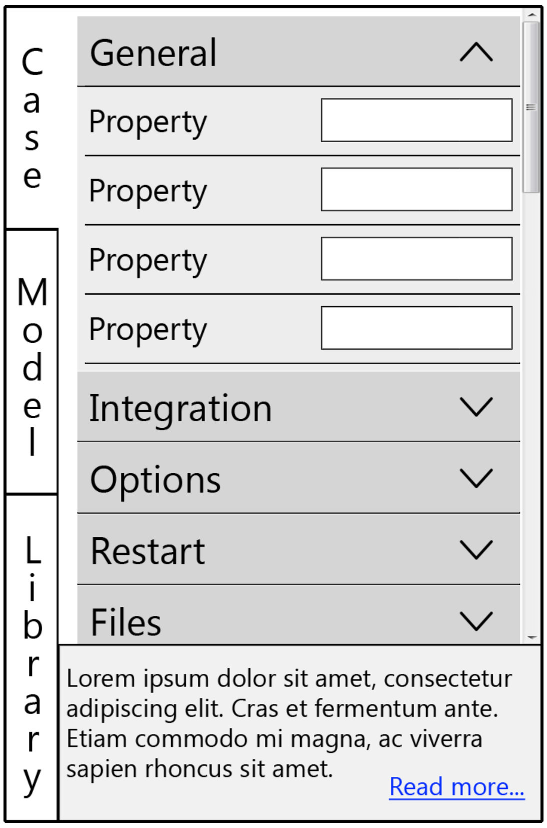
The groupings consists of properties with three alternative input methods, a simple text input for properties requiring text input, numerical inputs with units include a small dropdown list next to the text input. Additionally, properties requiring a predefined keyword can also be included, with a dropdown list directly in the entry. Properties that contains tabular data, or multiple values (lists) uses a link, which should launch a separate dialog with an interface suited for the type of input.

The property states available are optional, required and disabled. Optional entries will always be use a black color, required entries uses a red color and an asterisk to indicate the state. Lastly, disabled properties are shown with a lighter color and are intended to be locked and not respond to pointer clicks, and be silently skipped when tabbing.

Accessing the category tabs with a keyboard is solved with access keys, and the intended function is that up and down arrow keys can be used to switch between the tabs when a tab is focused. When a category tab is focused, the tab-key is used to switch focus to the list of property groupings and arrow keys and tab-key can be used to select and focus groups and properties.

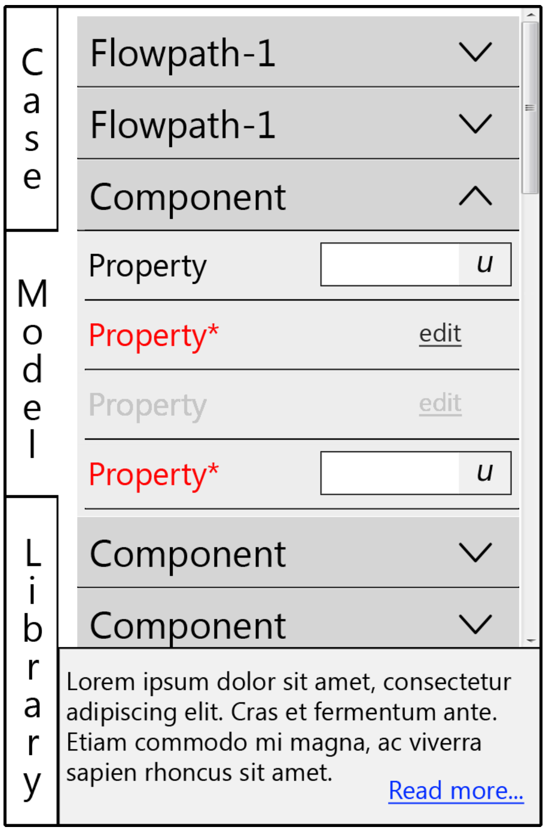
12.2.1 Design sketches

The designs presented on the next page use placeholder text for the descriptions located at the bottom in the case and model category-tabs.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras et fermentum ante. Etiam commodo mi magna, ac viverra sapien rhoncus sit amet.

Figure 12-2: property editor with case tab active



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras et fermentum ante. Etiam commodo mi magna, ac viverra sapien rhoncus sit amet.

Figure 12-3: property editor with model tab active

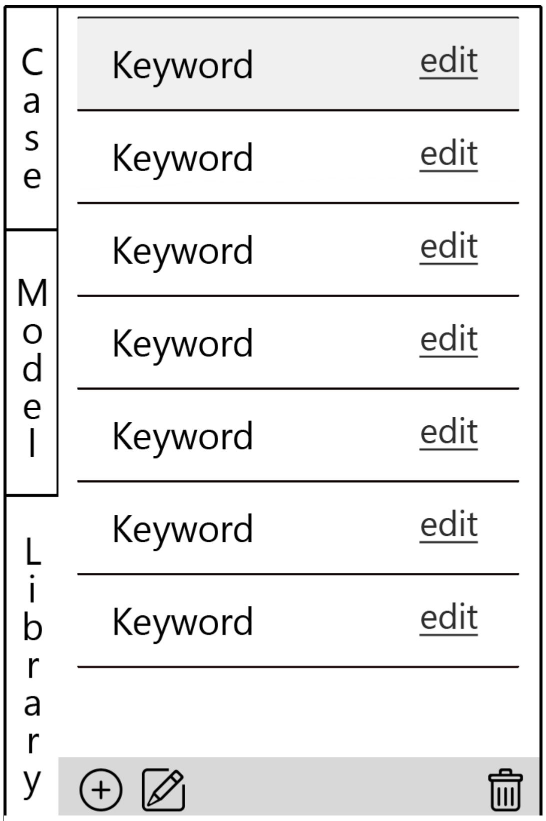


Figure 12-4: property editor with library tab active

12.3 Plotting interface

The proposed changes to the plotting interface are moving the legend to a window that can be docked to one of the edges and set to float. The intention is that the chart is resized automatically according to the state and position of the legend. If the legend is docked on one of the horizontal edges then the chart is resized horizontally and vertically when the legend is docked on the vertical edges. When the legend is floating, the chart will have more space available and the legend can be moved around freely on the screen or to a separate screen in multi-monitor environments. Other modifications include providing the title label as a control that can be manually edited without a separate dialog. Additional modifications to the variables listed in the legend include adding a min and max value field, so that the upper and lower boundaries can be visually confirmed. The legend window in docked mode is shown in Figure 12-5 and the legend window in floating mode is shown in Figure 12-6.

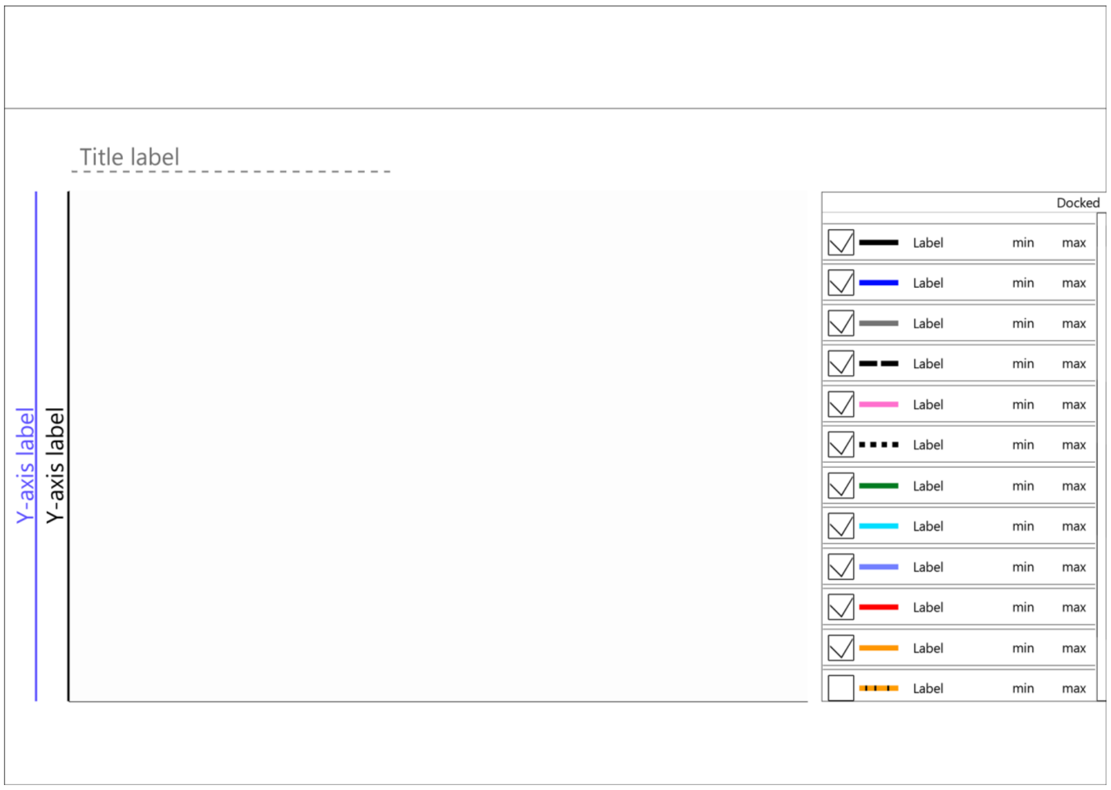


Figure 12-5: Plotting interface with docked legend window

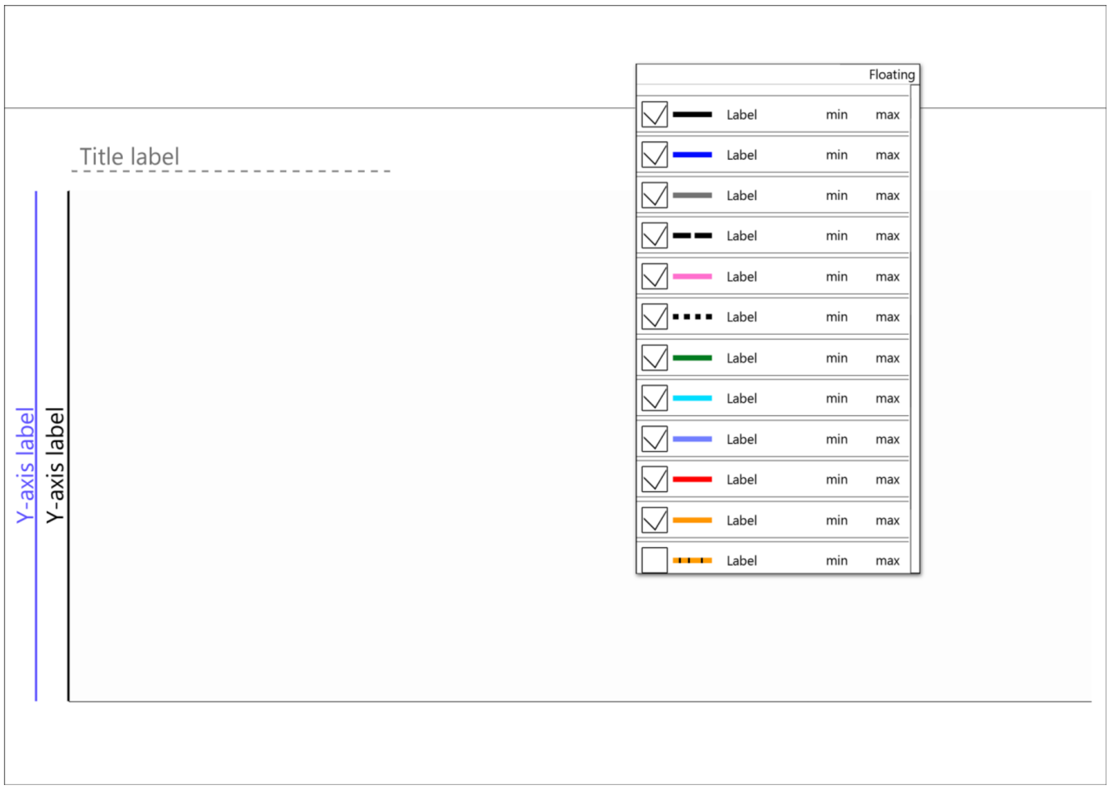


Figure 12-6: plotting interface with floating legend window

The concept designs above assume that the plotting interface is interactive and separate from the plot that is exported. The modifications include additional possibilities, such as, highlighting a variable in the legend that highlights the corresponding graph line, removing variables from the plot with a context menu without opening the variable selector. New variables could be created by highlighting multiple variables and perform calculations on the highlighted variables.

13 Discussion

During this project, we have investigated usability, design, and existing literature. Additionally, we have collected qualitative data from existing users, and created untested concept designs to provide a visual alternative for some aspects of the interface, which may come in use. The system was mainly evaluated in terms of the interface, and the system's requirements of its users.

The participants had similar workflows, the differences were primarily the use of different bundled tools and applications (such as Excel) to manage input due to stability issues. For example, one participant requested more available image formats when exporting plots to provide better support for workflows that include LaTeX. Moreover, a demonstrated workflow used Excel to format inputs when using the *pipeline editor* since the editor would freeze when working with large amounts of data.

The group session revealed some differences in opinions, one suggestion was to completely remove the plotting functionality, and replace it with an interface that would make it easier to export data to other applications. The other participants challenged the removal of plotting functionality, but were positive with the idea of making it easier to export data. Another suggestion was to add an additional tab for spreadsheets, which was initially protested by some participants, since it was seen as unnecessary to reinvent Excel functionality. However, the suggestion received more support, after the participant specified that the spreadsheet would only present and make it easier to copy data. Furthermore, it was observed that the more experienced participants frequently discussed functionality they deemed inconvenient or unnecessary time consuming, while less experienced had a higher focus on knowledge support in addition to convenience.

The data gathering revealed factors important to the participants where some were explicitly revealed and some inferred. The most basic, and maybe obvious, factor concerns whether the system provides the necessary functionality to achieve an overall goal (build and define a complete case), or complete a specific task (defining a specific condition). This report has no basis to claim that all the necessary functionality is available or specifically what is missing, but does have an example of a situation where a participant required specific components, that are not included in OLGA. In the situation, the participant needed to simulate a *nitrogen lift*[1] and required some specific components, such as coiled tubing[2] that was not available in OLGA. The participant, however, was able to use existing components and functionality to recreate the necessary behavior for a nitrogen lift, which illustrates that *flexibility* is needed and that the system does provide some flexibility. Additional factors include *stability*, *efficiency*, *knowledge*, *errors* and *reliability*.

Stability – Cases have a tendency to grow very large and complex, requiring a lot of time to build and define. If the system starts slowing down to inoperable levels or crashes, it is at the very least frustrating for the user and may result in loss of work and time spent. Additionally, even if there is no data loss it will still take some time to restart the system and returning to the previous state, and still risk experiencing the same issue with stability.

Efficiency – Time is important, for both the consultant and the client. In context of OLGA, efficiency is a multifaceted factor, when building and defining a case the fastest and safest procedures are desired to obtain more time for analyzing the results. When starting OLGA or loading a case, project or tool, likewise with loading simulation results, the user has better things to do than wait for it to complete, a few seconds can become excruciatingly long when it is needed "now".

Knowledge – Due to the complexity and the domain OLGA is used in, knowledge is important. To be able to accurately handle and analyze the results, the user must understand, or be able to obtain the needed knowledge. Inaccuracies due to lack of knowledge, may become expensive both in time. A weeklong simulation may need to be restarted due to a knowledge-based error, or may result in inaccurate reporting that may be used to guide the construction of a production system. In a worst case scenario can result in unforeseen stops in operation and damage to expensive equipment.

Errors – Reducing both technical and user errors are important, some types of errors may only result in inconveniences where the error is easy to fix though it requires effort. For example, recreating a plot after the plot was accidentally removed. Other errors may be more serious and require more time and effort to resolve, such as fixing the case definition

[1] A procedure where nitrogen is circulated into the system to displace liquids ("nitrogen lift," 2016)
[2] http://www.glossary.oilfield.slb.com/Terms/c/coiled_tubing.aspx

and rerunning the simulation. Furthermore, errors may impede stability, efficiency and reliability.

Reliability – This factor comes in two forms *reliability of the system* and *reliability of the results*. System reliability concerns the safety of the work of the users, such as if the user can rely on the system, not to lose data and work in case of unforeseen circumstances, functionality to not perform unwanted modifications or that destructive actions can be reverted. Result reliability concerns the accuracy of the case and simulation results, production systems produces high value during operation meaning that errors and inaccuracies can result in high loss of value.

The factors discussed above provide clues on how usability can be operationalized to better suit the situation with OLGA.

The conclusion from the interviews is that the further development of OLGA should continue with emphasis on stability improvements, and should consider providing more user control when it comes to loading large amounts of simulation results. Furthermore, the interface and documentation could be matched more closely, which was pointed by the interview subjects as outdated and vague in some areas. Furthermore, documentation that is more accessible for novice users is an area that should be considered for improvement. In this perspective, more effort should be dedicated so that to improve the documentation for knowledge support, and to split it into multiple parts with focus on different aspects, such as, completion of common tasks, introduction and in-depth details on concepts. The three mentioned aspects would provide the user with a guide on how the interface can be used to solve tasks, gain an introductory knowledge of the various concepts without being overwhelmed with technical details.

Additionally, building and defining cases are time-consuming processes involving a lot of *fiddling* where the user navigates back and forth between the model and model browser, as well as, within the model browser itself between components in the navigation pane and properties in the property editor. This factor sparked the idea for an alternative concept for a property editor, which intends to focus the effort into a specific area, such as when building a model only the properties related to the model should be visible, while properties for output and conditions should be either removed or hidden.

Handling the simulation results can be improved. The OLGA interface gives an impression that all of the simulation results are to be used inside OLGA and that data visualizations are the only end-product needed. However, while the former may be true for some user groups, it was observed that the participants frequently required ability to export selective parts of data into different common formats. The current data export functionality is cumbersome and requires editing and removing data manually. Regarding the charts, OLGA could take more advantage of being an interactive GUI. The interface could provide direct editing of labels, instead of requiring interaction through dialogs. Furthermore, OLGA could provide differentiated plots that are optimized for inclusion within reports and presentations and plots that are optimized for real-time interaction during simulation settings.

The project has been challenging in multiple aspects, such as the various approaches to usability, which unintentionally introduces a challenge in identifying the *operationalization* of usability in literature. The literature regarding usability in technical interfaces has been difficult to locate, usability studies are mainly focused on web-interfaces or interfaces for a more mainstream audience, and for systems with less technical complexity. Studies targeting or developing interfaces for technical systems, has frequently included usability as a *feature* for the system without indicating the meaning of usability in the context. Additionally, many studies targeting engineering and simulation systems operate with the word *usability* to indicate how usable the product, created with the system, is for real scenarios.

Furthermore, this study has highlighted some areas that may require more research specifically targeting data extraction and visualization, the different user groups within the global user group, and how different entities applies OLGA as a tool within their workflow. Alternative visualization methods could be relevant to investigate, such as how to visualize propagation in the production system. Additionally, it may be necessary to perform comparative studies with similar systems. The studies could focus on highlighting differences between the systems and the user performance in areas, such as, building a model and defining conditions, as well as, the visualization method, and analysis of simulation results. The studies may require familiarity with the domain and both systems. An area that should be focused on is learning more about the target user group, OLGA is used internationally and it may be beneficial to know if any user groups require a more specialized interface, as well as, how the use of OLGA varies among user groups according to experience, knowledge, education, training and culture.

Some observations about the techniques used to gather information is also worth mentioning. The techniques were adapted during the project to account for delays and other circumstances, which resulted in contextual interviews being held in meeting rooms, and individual co-creation sessions being replaced with a focus group with tools to make notes and sketches. Contextual interviews are suited to gather information about individual workflows and discover how the system is used to achieve goals and solve specific tasks. However, with contextual interviews it was difficult to establish how the domain knowledge was applied and in general difficult to achieve a better understanding of the domain. Focus groups was more suited to discover the differences in how the participants handled the system and domain knowledge, and provided a clearer understanding of the domain. Additionally, with a group the participants pushed each other by raising topics that were unknown to the interviewer, such as the desire for plotting functionality that visualizes propagation in the flow paths, which most likely would not have been raised by the interviewer. Thus, the observations suggests that a focus group should be held early in the process, and should be seriously considered when working with a complex and unfamiliar domain as it may provide more insight. While contextual interviews are more suited to gather, information that is more detailed.

14 Conclusion

Few usability studies focus on user interface design for complex and domain-specific systems. We are currently preparing a research article to be submitted to a conference in usability that documents the results of this study. We believe the findings can be beneficial to practitioners and stakeholders working with interfaces for complex interfaces.

We recommend that the future development of OLGA continue with emphasis on improvements to stability, reducing user errors and improving user control. We also suggest that implementing functionality for exporting data and automating tasks should be considered. Additionally, the documentation should be updated and improved with emphasis on knowledge support.

Future work should focus on identifying the various use cases and user groups that implement OLGA in their workflow, in order to discover potential improvements. The data visualization used in the current system are line charts and future work should consider exploring, alternative methods for visualizing simulation results and exporting data. Additionally, performing comparative usability studies of multiple flow assurance systems would complement the existing literature that is focused on the technical performance of similar systems.

15 References

Airam Sausen, Paulo Sausen, & Mauricio de Campos. (2012). *The Slug Flow Problem in Oil Industry and Pi Level Control*. In Dr. Jorge Salgado Gomes (Ed.). Retrieved from http://www.intechopen.com/books/new-technologies-in-the-oil-and-gas-industry/the-slug-flow-problem-in-oil-industry-and-pi-level-control

Aiyejina, A., Chakrabarti, D. P., Pilgrim, A., & Sastry, M. K. S. (2011). Wax formation in oil pipelines: A critical review. *International Journal of Multiphase Flow, 37*(7), 671-694. doi:http://dx.doi.org/10.1016/j.ijmultiphaseflow.2011.02.007

Arning, K., Himmel, S., & Ziefle, M. (2016). “Overloaded, Slow, and Illogical” A Usability Evaluation of Software for Product Manufacturing Processes with a Special Focus on Age and Expertise of CAM Users.

Bevan, N. (1995). Measuring usability as quality of use. *Software Quality Journal, 4*(2), 115-130. doi:10.1007/BF00402715

Bevan, N. (2001). International standards for HCI and usability. *International Journal of Human-Computer Studies, 55*(4), 533-552. doi:http://dx.doi.org/10.1006/ijhc.2001.0483

Bevan, N. (2006). International standards for HCI. *Encyclopedia of human computer interaction, 362*.

Bevan, N. (2009). *What is the difference between the purpose of usability and user experience evaluation methods.* Paper presented at the Proceedings of the Workshop UXEM.

Bevan, N., Kirakowski, J., & Maissel, J. (1991). *What is usability?* Paper presented at the Proceeding of the Fourth International Conference on HCI.

Beyer, H., & Holtzblatt, K. (1997). *Contextual design: defining customer-centered systems*: Elsevier.

Bhasin, R., & Venkata, P. P. K. (2009). Development of a Cross-Platform Framework to Assist Computational Fluid Dynamics (CFD) Users. *CURIE Journal, 2*(1), 38-48. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=bsh&AN=39775347&site=ehost-live

Brooke, J. (1996). SUS-A quick and dirty usability scale. *Usability evaluation in industry, 189*(194), 4-7.

Carducci, F., Del Monaco, A., Giacchetta, G., Leporini, M., & Marchetti, B. (2015). Development and application of an innovative tool to automate the process of results extraction from the thermo-hydraulic simulator Olga. *Petroleum, 1*(2), 164-168. doi:http://dx.doi.org/10.1016/j.petlm.2015.06.003

Chou, E. (2002). *Redesigning a large and complex website: how to begin, and a method for success*. Paper presented at the Proceedings of the 30th annual ACM SIGUCCS conference on User services, Providence, Rhode Island, USA.

Cockton, G., & Lavery, D. (1999). *A framework for usability problem extraction.* Paper presented at the Proc Interact.

Dickinson, A., Eisma, R., & Gregor, P. (2002). Challenging interfaces/redesigning users. *SIGCAPH Comput. Phys. Handicap.*(73-74), 61-68. doi:10.1145/960201.957217

Experience Dynamics. (2015). Science of Usability. Retrieved April 20, 2015 from https://www.experiencedynamics.com/approach/science-usability

Følstad, A., Law, E., & Hornbæk, K. (2012). *Analysis in practical usability evaluation: a survey study*. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Austin, Texas, USA.

Gupta, D., Ahlawat, A., & Sagar, K. (2014, 27-29 Nov. 2014). *A critical analysis of a hierarchy based Usability Model.* Paper presented at the Contemporary Computing and Informatics (IC3I), 2014 International Conference on.

Heuristic Evaluation. (2007). Retrieved May 15, 2015 from http://www.usabilitybok.org/heuristic-evaluation

IEEE. (1990). 610.12-1990 *IEEE Standard Glossary of Software Engineering Terminology* (pp. 1-84): Institute of Electrical and Electronics Engineers.

IEEE. (2015). Purpose of Standards Education. Retrieved May 1, 2015 from https://www.ieee.org/education_careers/education/standards/why.html

IFE. (n.d). The history of OLGA. Retrieved May 12, 2015 from http://www.ife.no/en/ife/departments/process_and_fluid_flow_tech/historienomol ga

Interviews. (2006). Retrieved May 15, 2015 from http://www.usabilitynet.org/tools/interviews.htm

ISO. (1998). 9241-11:1998. *Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on usability*.

ISO, & ISQS. (2011). ISO/IEC 25010:2011. *Systems and software engineering---Systems and software Quality Requirements and Evaluation (SQuaRE)---System and software quality models*.

ISO/IEC. (2001). ISO/IEC 9126-1 Standard, Software Engineering, Product Quality, Part 1: Quality Model.

Johnson, C. M., Johnson, T. R., & Zhang, J. (2005). A user-centered framework for redesigning health care interfaces. *Journal of Biomedical Informatics, 38*(1), 75-87. doi:http://dx.doi.org/10.1016/j.jbi.2004.11.005

Kayisli, K., Tuncer, S., & Poyraz, M. (2013). An educational tool for fundamental DC–DC converter circuits and active power factor correction applications. *Computer Applications in Engineering Education, 21*(1), 113-134.

Lado, M. J., Méndez, A. J., Roselló, E. G., Dacosta, J. G., Pérez-Schofield, J. B. G., & Cota, M. P. (2006). R-Interface: An alternative GUI for MATLAB. *Computer Applications in Engineering Education, 14*(4), 313-320.

Logan, R. J. (1994). Behavioral and emotional usability: Thomson Consumer Electronics.E. W. Michael (Ed.), In *Usability in practice* (pp. 59-82): Academic Press Professional, Inc.

Maguire, M., & Bevan, N. (2002). User Requirements Analysis.J. Hammond, T. Gross, & J. Wesson (Eds.), In *Usability* (Vol. 99, pp. 133-148): Springer US.

Mao, J.-Y., Vredenburg, K., Smith, P. W., & Carey, T. (2005). The state of user-centered design practice. *Commun. ACM, 48*(3), 105-109. doi:10.1145/1047671.1047677

Marczak, M., & Sewell, M. (n.d). Using Focus Groups for Evaluation. Retrieved May 15, 2015 from http://ag.arizona.edu/sfcs/cyfernet/cyfar/focus.htm

Mayhew, D. J. (1999). *The usability engineering lifecycle.* Paper presented at the CHI'99 Extended Abstracts on Human Factors in Computing Systems.

McDonald, S., Monahan, K., & Cockton, G. (2006). *Modified contextual design as a field evaluation method*. Paper presented at the Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles, Oslo, Norway.

McNamara, N., & Kirakowski, J. (2005, 10-13 July 2005). *Defining usability: quality of use or quality of experience?* Paper presented at the Professional Communication Conference, 2005. IPCC 2005. Proceedings. International.

Microsoft. (2015). Guidelines (Windows). Retrieved May 22, 2015 from https://msdn.microsoft.com/en-us/library/windows/desktop/dn688964

Nielsen, J. (1993). Iterative User-Interface Design. *Computer, 26*(11), 32-41. doi:10.1109/2.241424

Nielsen, J. (1995). 10 Usability Heuristics for User Interface Design. Retrieved April 23, 2015 from http://www.nngroup.com/articles/ten-usability-heuristics/

Nielsen, J. (2012). Usability 101: Introduction to Usability. Retrieved May 10, 2015 from http://www.nngroup.com/articles/usability-101-introduction-to-usability/

nitrogen lift. (2016). Retrieved May 09, 2016 from http://www.glossary.oilfield.slb.com/Terms/n/nitrogen_lift.aspx

Norman, D. A. (1983). Some observations on mental models. *Mental models, 7*(112), 7-14.

Norman, D. A. (1988). *The psychology of everyday things*: Basic books.

Participatory Design. (2005). Retrieved May 16, 2015 from http://cpsr.org/issues/pd/

Richardson, K. H. (2013). It's not about usability. *Commun. Des. Q. Rev, 1*(3), 54-56. doi:10.1145/2466489.2466500

Rose, A., Shneiderman, B., & Plaisant, C. (1995). *An applied ethnographic method for redesigning user interfaces*. Paper presented at the Proceedings of the 1st conference on Designing interactive systems: processes, practices, methods, & techniques, Ann Arbor, Michigan, USA.

Roser, T., Samson, A., Humphreys, P., & Cruz-Valdivieso, E. (2009). Co-Creation: New Pathways to Value: An Overview. *Promise & LSE Enterprise*.

Sanders, E., & Simons, G. (2009). A Social Vision for Value Co-creation in Design. Retrieved May 22, 2015 from http://timreview.ca/node/310

Sanders, E., & Stappers, P. J. (2008). Co-creation and the new landscapes of design. *CoDesign, 4*(1), 5-18. doi:10.1080/15710880701875068

Schlumberger. (2012). Schlumberger Announces Agreement to Acquire SPT Group. Retrieved May 12, 2015 from https://www.slb.com/news/press_releases/2012/2012_0320_slbsptgroup_pr.aspx

Schlumberger Software (Producer). (2015, February 13). PIPESIM and OLGA Simulators: Flow Assurance from concept to Operations. Retrieved from https://www.youtube.com/watch?v=857wSFqZGSU

Scholtz, J. (2004). Usability evaluation. *National Institute of Standards and Technology*.

Seffah, A., Donyaee, M., Kline, R., & Padda, H. (2006). Usability measurement and metrics: A consolidated model. *Software Quality Journal, 14*(2), 159-178. doi:10.1007/s11219-006-7600-8

Shneiderman, B., Plaisant, C., & Cohen, M. (2009). *Designing the User Interface* (5 ed.): Prentice Hall.

Spinuzzi, C. (2005). The Methodology of Participatory Design. *Technical Communication, 52*(2), 163-174. Retrieved from http://www.ingentaconnect.com/content/stc/tc/2005/00000052/00000002/art00005

Spool, J. M. (2007). Field Studies: The Best Tool to Discover User Needs. Retrieved May 18, 2015 from http://www.uie.com/articles/field_studies/

Statoil. (2009). Slug Control. Retrieved November 15, 2015 from http://www.statoil.com/en/TechnologyInnovation/FieldDevelopment/FlowAssurance/SlugControl/Pages/default.aspx

StreamLine. (n.d). A Comparison between commercial Pipeline Fluid-dynamic Code OLGA vs. LEDAFLOW. Retrieved March 27, 2016 from www.stream-line.it/Banchmark-Olga-vs-Leda.pdf

Task analysis. (2006). Retrieved May 15, 2015 from http://www.usabilitynet.org/tools/taskanalysis.htm

Tine Bauck Irmann-Jacobsen, & Hægland, B. (2014). *Flow Assurance & Operability*. In MEK4450. Retrieved from http://www.uio.no/studier/emner/matnat/math/MEK4450/h14/undervisningsmateriale/module-4/mek4450_flowassurance_pensum.pdf

User Manual OLGA 7.3. (n.d). OLGA.

Villberg, A. (2007). *Design challenges of an ontology-based modelling and simulation environment.* HELSINKI UNIVERSITY OF TECHNOLOGY.

Vision Critical. (2015). Co-creation 101: How to use the crowd as an innovation partner to add value to your brand. Retrieved May 18, 2015 from https://www.visioncritical.com/cocreation-101/

Weyers, B., Burkolter, D., Luther, W., & Kluge, A. (2012). Formal Modeling and Reconfiguration of User Interfaces for Reduction of Errors in Failure Handling of Complex Systems. *International Journal of Human-Computer Interaction, 28*(10), 646-665. doi:10.1080/10447318.2011.654199

Wilding, C. (1998). *Practical GUI screen design: making it usable*. Paper presented at the CHI 98 Cconference Summary on Human Factors in Computing Systems, Los Angeles, California, USA.

Wright, P., McCarthy, J., & Meekison, L. (2005). Making Sense of Experience.M. Blythe, K. Overbeeke, A. Monk, & P. Wright (Eds.), In *Funology* (Vol. 3, pp. 43-53): Springer Netherlands.

Wu, W., & Ng, E. (2001). Accuracy and usability of daylighting simulation for designing buildings in urban sites.