

The Octave GUI

The screenshot displays the Octave GUI interface. At the top, the title bar reads "Octave". Below it is a toolbar with icons for file operations and navigation, and a "Current Directory" field showing `/home/jacob/octave/gui/bin`.

The interface is divided into three main panels:

- Workspace:** A table showing variables in the workspace.

Name	Type	Value
▼ Local		
a	scalar	100
b	scalar	2
Global		
Persistent		
▼ Hidden		
.nargin	scalar	0
- Command Window:** Displays the Octave startup message and user input/output.

```
GNU Octave, version 3.7.0+
Copyright (C) 2012 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type `warranty'.

Octave was configured for "x86_64-unknown-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.

For information about changes from previous versions, type `news'.

warning: function ./test.m shadows a core library function
:1> test
:2> █
```
- Current Directory:** A file browser showing the contents of the current directory.
 - test.m
 - octave-gui
 - octave-core

At the bottom, there is a "Command History" panel showing a list of previous commands like `test`, `exit`, `euler(1e-7)`, and `cd`. Below the panels are tabs for "Command Window" and "Octave Editor". At the very bottom, a status bar shows: "CPU time of the GUI in octave thread - generating rx events: ~0 ms - processing tx events: ~0 ms (0 events)".

What makes a good GUI?

- Requirements for a GUI are summarized in the EN ISO 9241 standard, which includes:
- Suitability
- Must be self-descriptive
- Accessibility (different means to perform a certain task)
- Adaptability
- Must meet the user's expectations

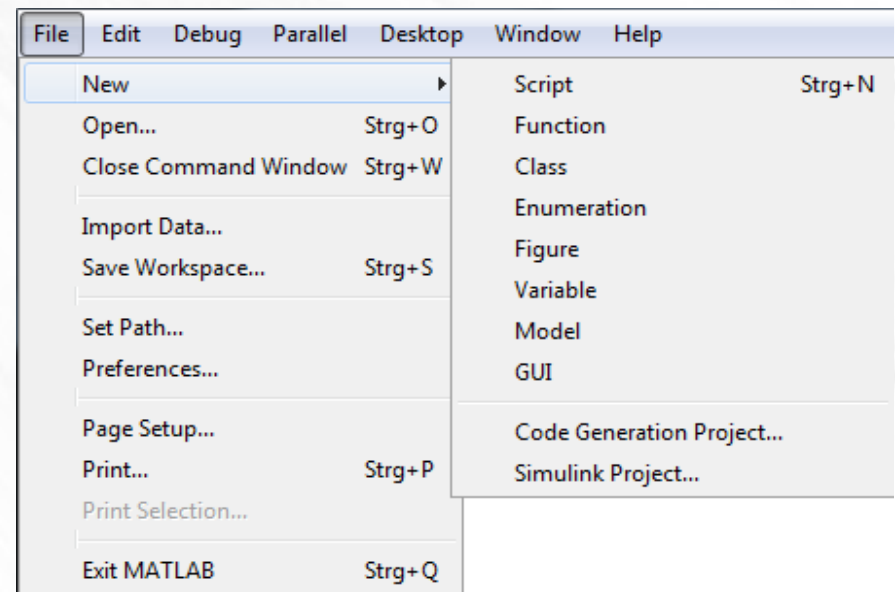
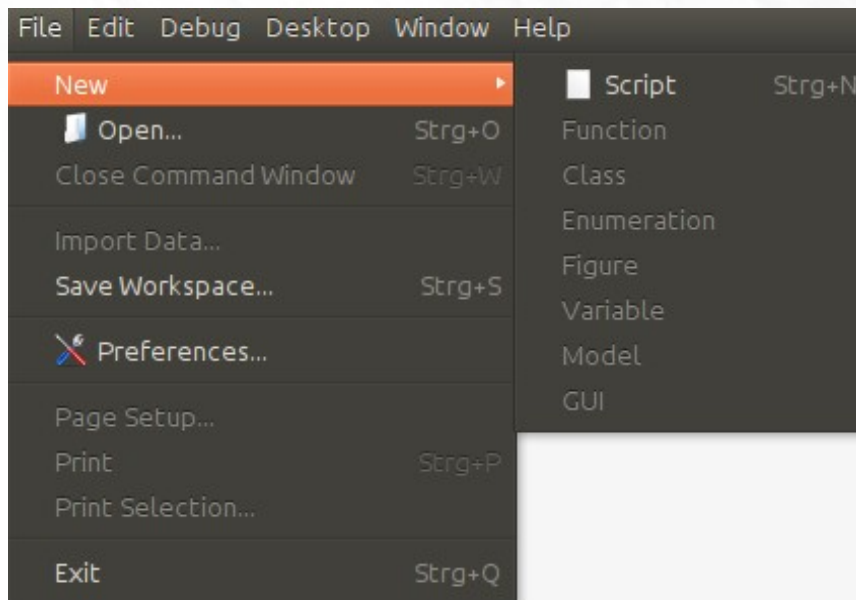
=> There is no **absolute measure** to determine whether a GUI is well-done or not. Whether the design of a GUI is good or not heavily depends on the target audience.



Taken from: <http://www.ssw.com.au/>

Design goals for the Octave GUI

- Provide an visual environment that is similar to that of MATLAB. Keep care of menu structure, shortcuts, naming conventions.
- Extend the visual environment wherever it makes sense to do that (for example: *History search*)



What is Qt?

- Qt is an *application development framework*
- An application development framework is a set of classes that helps developers to create feature-rich applications faster by providing common functionality on a very abstract level
- It is common to use application development frameworks in software development today
- Qt is free software licensed under the GPL

Design features of Qt

- Very high abstraction level
- Aim for platform-independency
- Suited for high-performance, multimedia applications
- Easy to learn, just a few repetitive concepts („learned it once, learned it all“)
- Very good documentation, integrated into the QtCreator IDE

Basic Concepts: QObject

- Much like in Java, Qt introduces a basic object classes for all non-trivial data structures: QObject
- By default, each instance of a QObject is unique, it cannot be copied
- Each QObject can benefit from a language extension Qt provides, called *signals and slots*

Basic Concepts: QObject

- Much like in Java, Qt introduces a basic object classes for all non-trivial data structures: QObject
- By default, each instance of a QObject is unique, it cannot be copied
- Each QObject can benefit from a language extension Qt provides, called *signals and slots*

Basic Concepts: Signals and Slots

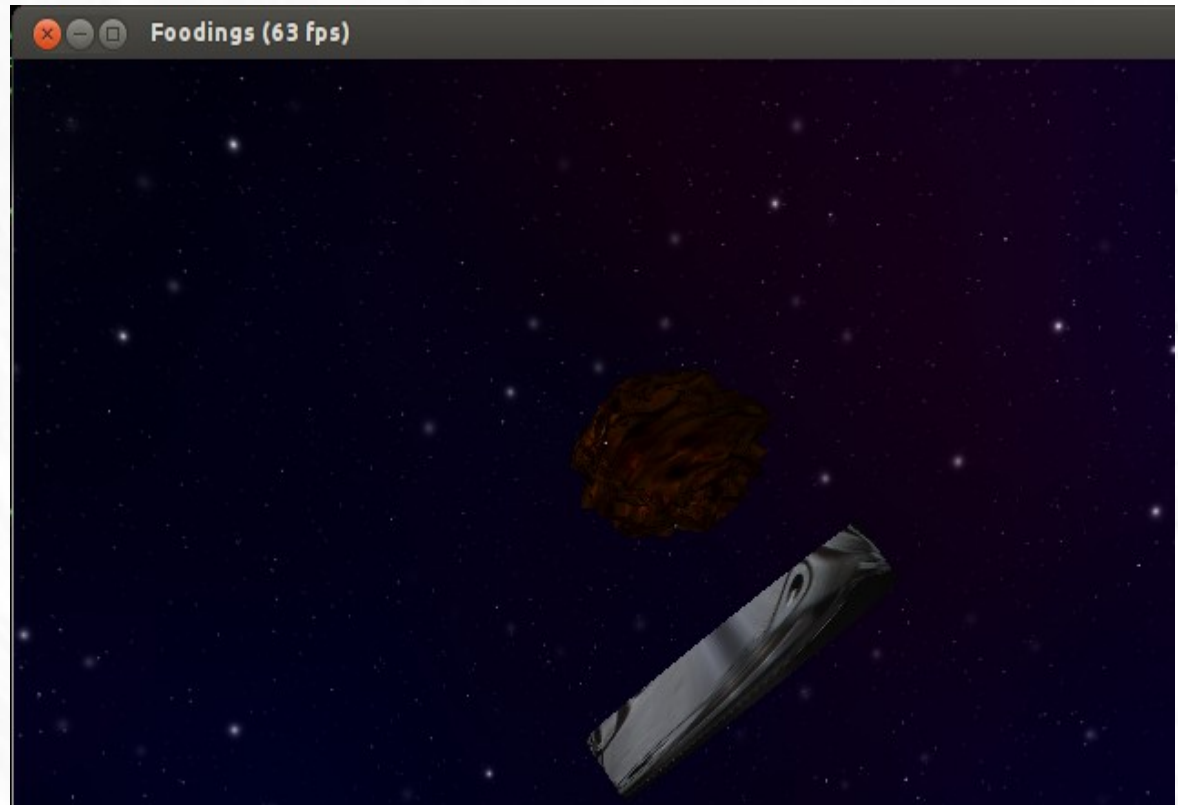
- In order to provide loose coupling of objects (that's a desirable feature in object-oriented programming), QObject can communicate via signals and slots
- Each QObject can *emit signals*, however, it does not have any knowledge about who listens to the signal and if it has an effect at all
→ „Fire and forget“
- Each QObject can offer slots. Slots can be connected and disconnected to signals **at runtime**, without the need of telling the QObjects about that

Basic Concepts: QWidgetets

- QWidget derives from QObject
- Widget = „Visual Gagdet“
- QWidgetets are usually used for displaying visual elements. All visual elements have to have QWidget as a base class
- QWidgetets can be „stacked“ one into another by explicitly stating a hierarchy via the *parent*-parameter in the QWidget-
constructor
- Once a parent has been assigned to a QWidget, the parent cares for cleaning up memory (ie. deleting child widgets will lead to a segfault)
- QWidget without a parent are top-level-widgets. The user has to take for cleaning up memory.

Basic Concepts: Neatless integration of OpenGL

- With QGLWidget, Qt integrates OpenGL neatly. Qt can manage several OpenGL displays at the same time.



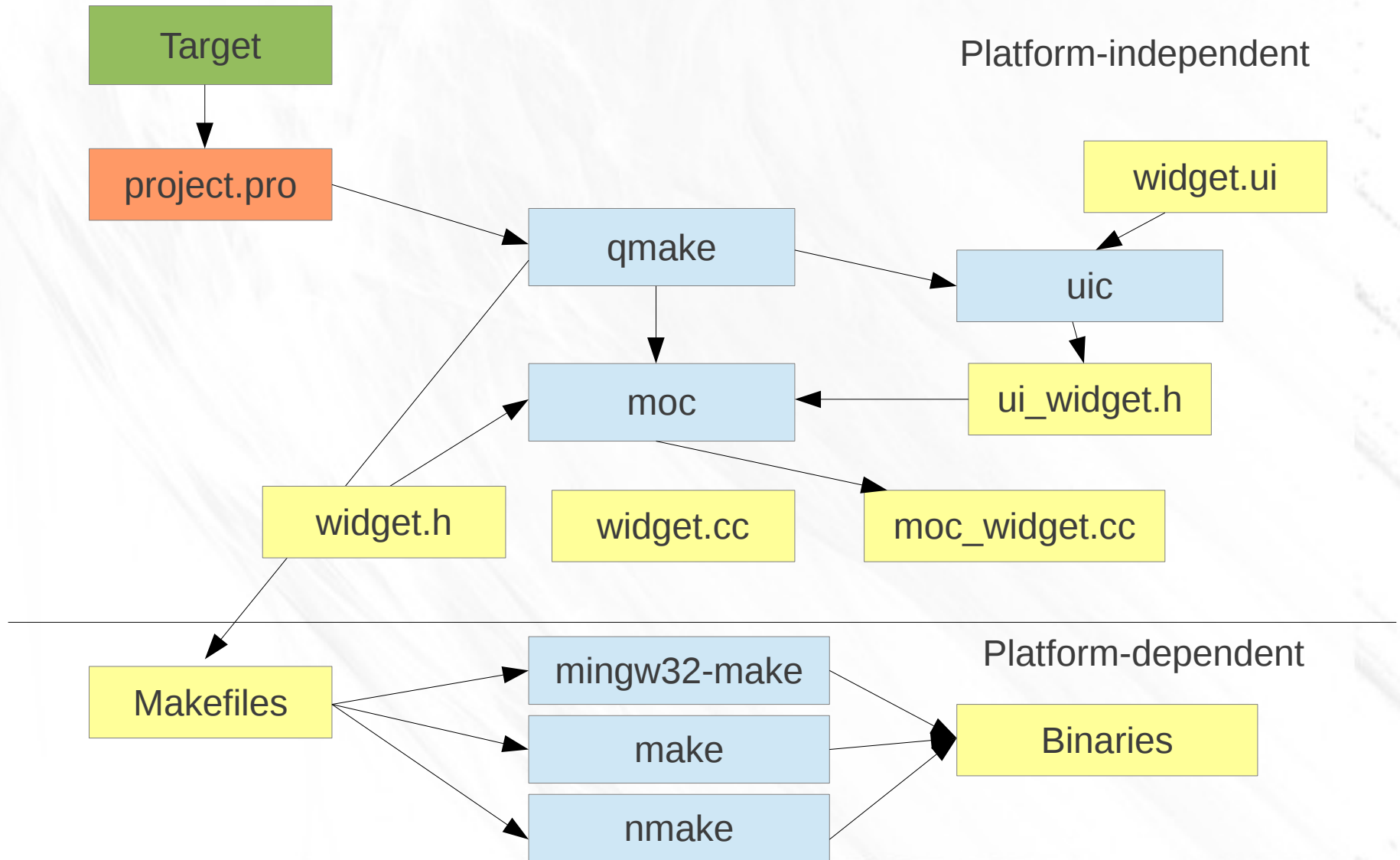
Basic Concepts: Layouting and Styling

- QWidgets can be aligned in horizontal and vertical layouts
- The actual size size is determined by the size policy of each QWidget
- Layouts can be cascaded to provide any widget layout that is desired
- QWidgets can be styled like webpages with regular CSS

Basic Concepts: Qt Devel Tools

- `qmake`: Reads a *.pro-project file and generates makefiles for various „targets“ (platforms). `qmake` is able to generate project files for other IDEs, like for example Visual Studio.
- `moc`: Meta-Object-Compiler. This tool is used to parse source files containing QObjects in order to generate glue code for the signals and slots system.
- `uic`: User-Interface-Compiler. Reads XML-user interface definitions and creates C++ source code.
- `lupdate`: Part of Qt's language translation system. Updates the language files by parsing the sources for the `tr(..)` macro.
- `lrelease`: Compiles language files that can be shipped with and loaded by Qt applications.

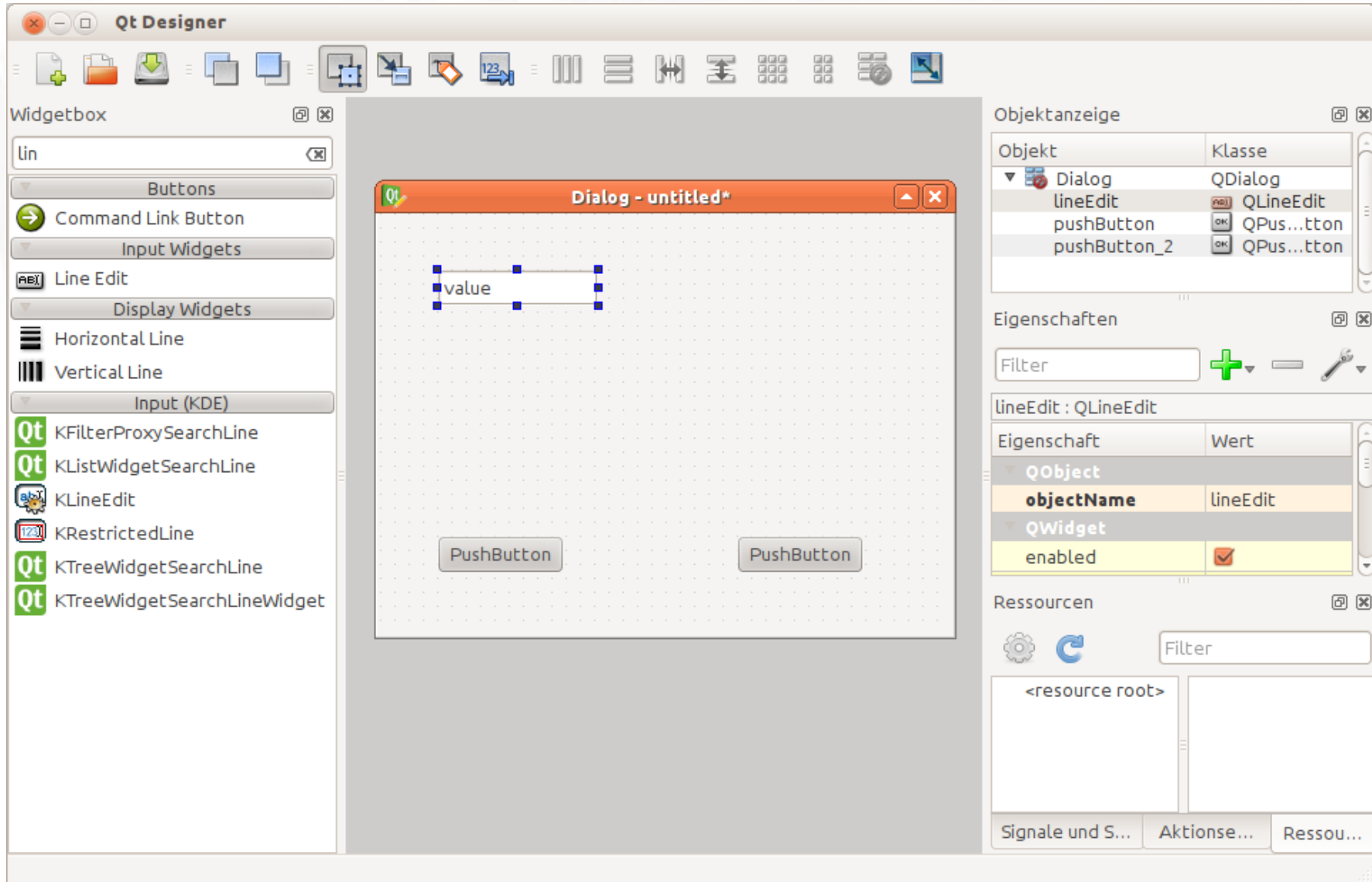
Minimal Qt Build Process



Designing a GUI with Qt

- Qt offers a WYSIWYG editor for designing GUIs: QtDesigner
- Qt offers a pretty neat system for layouts and size policies
- Once a GUI has been designed, it will store the result in an XML-format file (.ui-file)
- The uic-tool generates header files, these will contain a widget class that can be subclassed and extended with C++ code
- - => **Combining the best of two worlds:** *Code generation* for designing purposes, *writing code* in order to write detailed GUI logic and implementing extended features.
 - => Code generation and writing code do not interfere.

QtDesigner



Localization

- Qt has built-in localization support. Within Qt, all user interface strings have to be surrounded with *tr(...)*
- With the `qupdate`-command the language input files for the specified languages will be updated.
- QtLinguist can be used for translators to do translation within an easy to learn IDE.
- With `qrelease` the language files get compiled into distributable „language packages“ that can be loaded by Qt applications.

There has already been a few translations for the GNU Octave GUI.

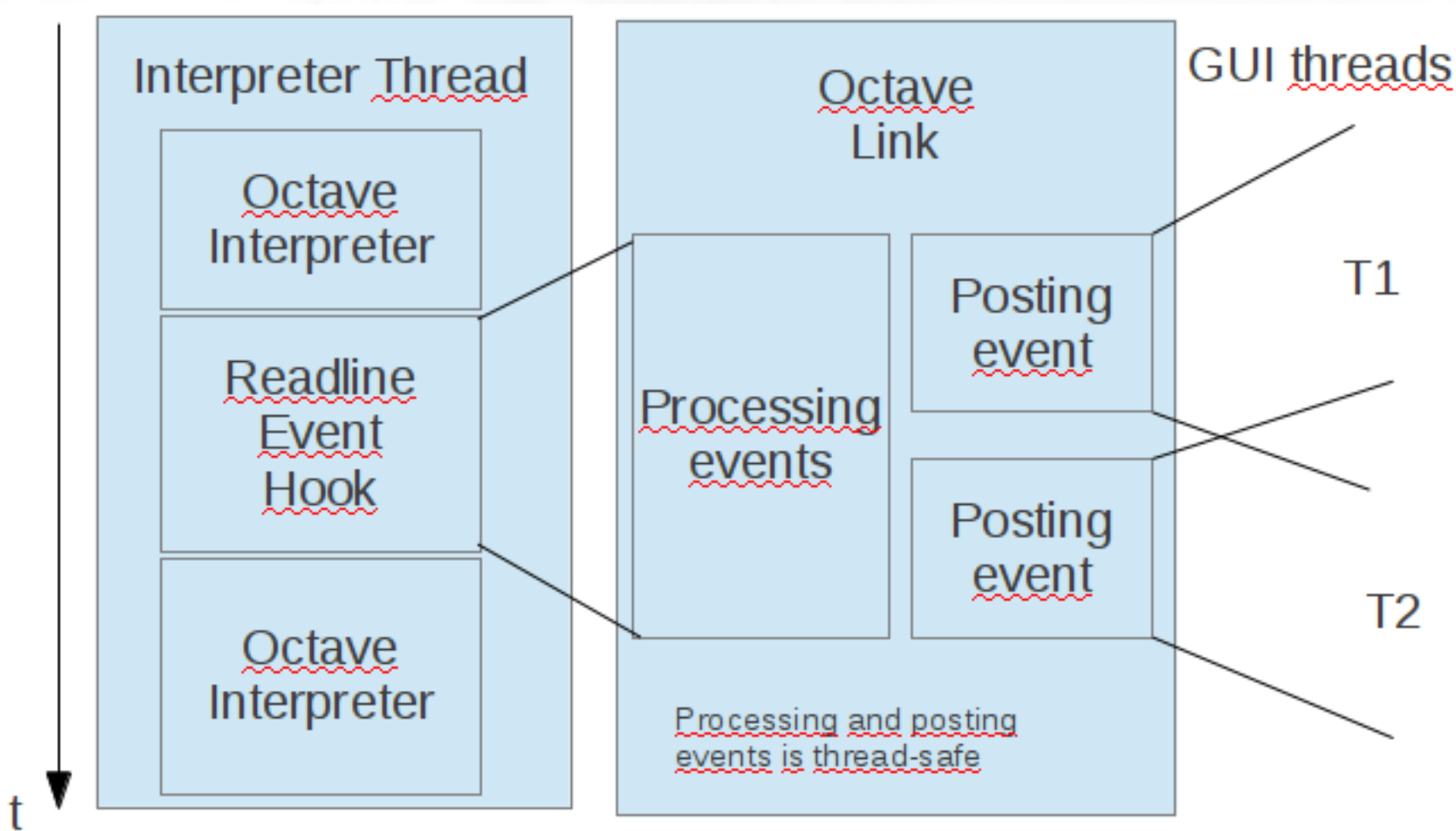
GNU Octave and thread-safety

- Problem: GNU Octave itself is not thread-safe, but applications with GUIs have to be multithreaded in order to keep the GUI responsive.
- Solution 1: Make GNU Octave threadsafe.
 - + Often, GNU Octave can benefit heavily from systems that are able to execute multiple threads in parallel in hardware
 - + Applying a GUI would be easy
 - Nearly impossible to achieve due to lots of global state in GNU Octave
 - Performance decrease for tasks that cannot be parallelized or hardware that does not support multiple threads in hardware

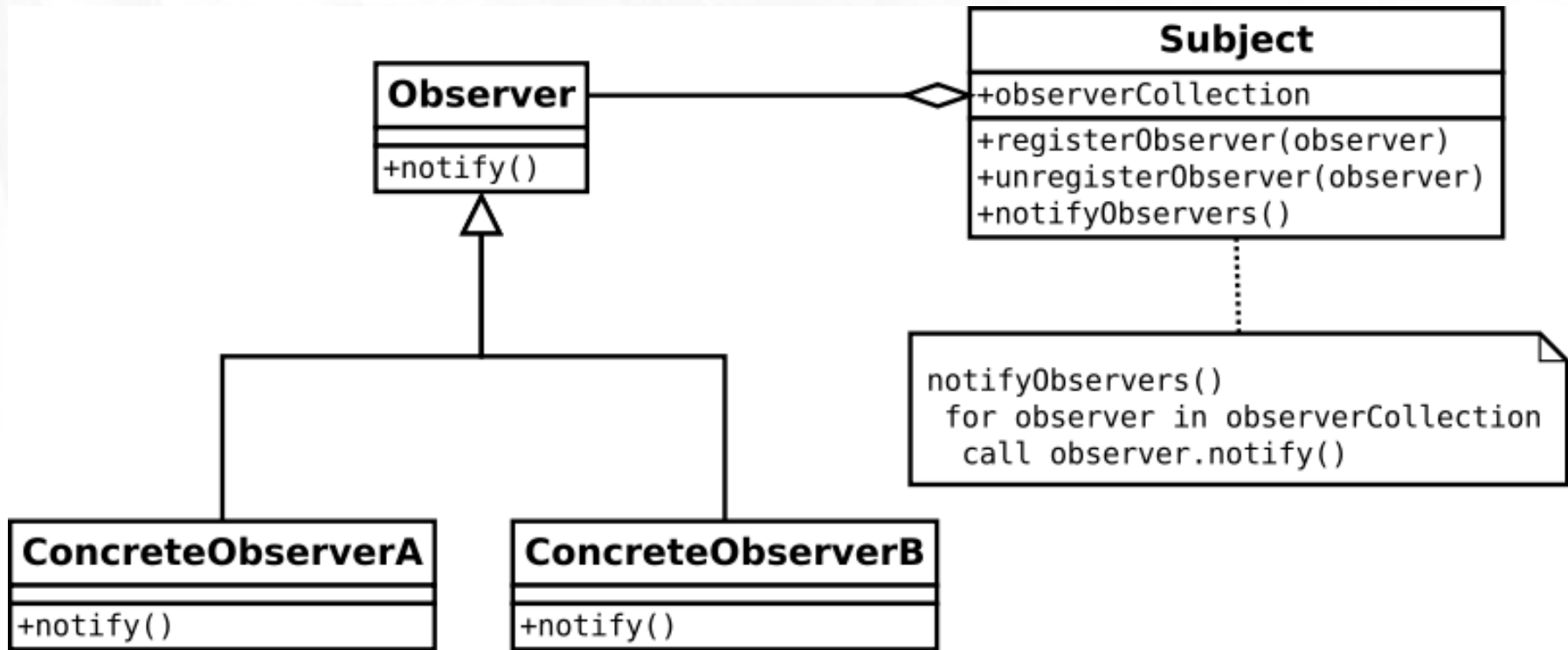
GNU Octave and thread-safety

- Problem: GNU Octave itself is not thread-safe, but applications with GUIs have to be multithreaded in order to keep the GUI responsive.
- Solution 2: Serialize communication between GNU Octave and other threads.
 - + Minimal-invasive approach, no modification of former GNU Octave code
 - + Easy to implement and understand
 - Performance decrease for all tasks, since additional time in the GNU Octave thread will be spent. *Possible solution → Use the GUI for interactive research and code development. Once this has been done, use the GNU Octave CLI version to run your scripts on a larger scale.*

Event-based communication



Events implemented with Observer-Pattern



Source: <http://en.wikipedia.org/wiki/File:Observer.svg>

Integrating the command line

- Former GUIs implemented a pipe-based communication by launching GNU Octave as a subprocess.

Disadvantages:

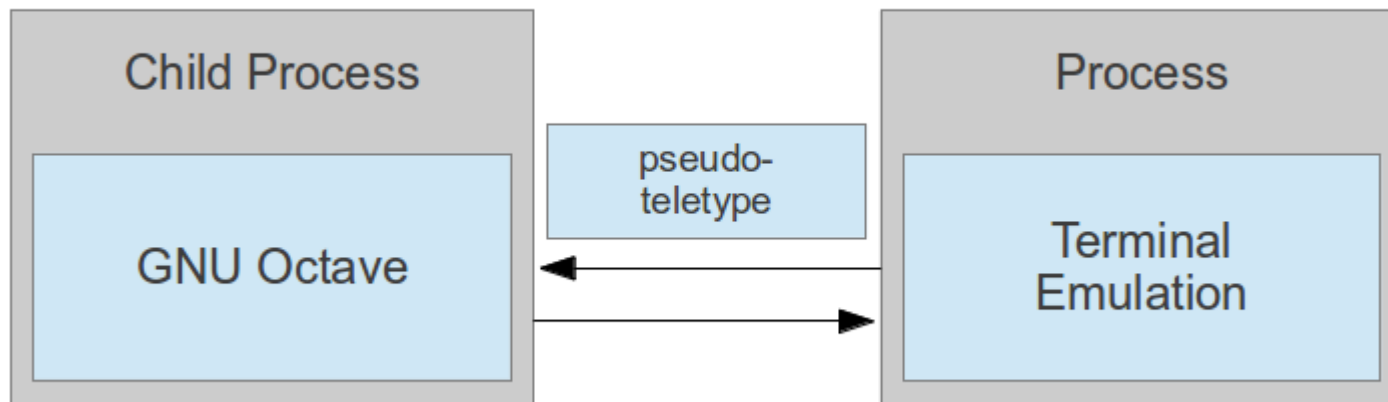
- The GUI has to parse GNU Octave's output and generate input → slow and error-prone
- The GUI has no access to internal data → limited functionality
- The GUI has to reimplement basically all functionality that is already given by readline

GNU Octave GUI's approach

- Based on code from KDE's Konsole: **Full terminal emulation**. On Windows: Console2 approach.
- Theoretically, offers the same functionality as GNU Octave run in a terminal window, with no compromises on functionality
- John Swensen first came up with this idea in his GTK+/Qt *OctaveDE*.

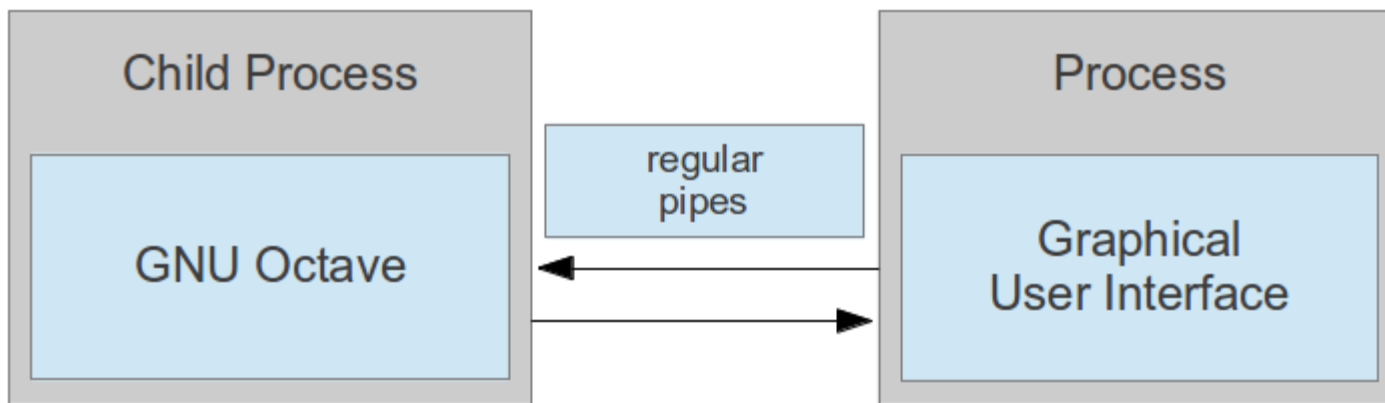
Different ways of running octave

a) GNU Octave running in a terminal emulation



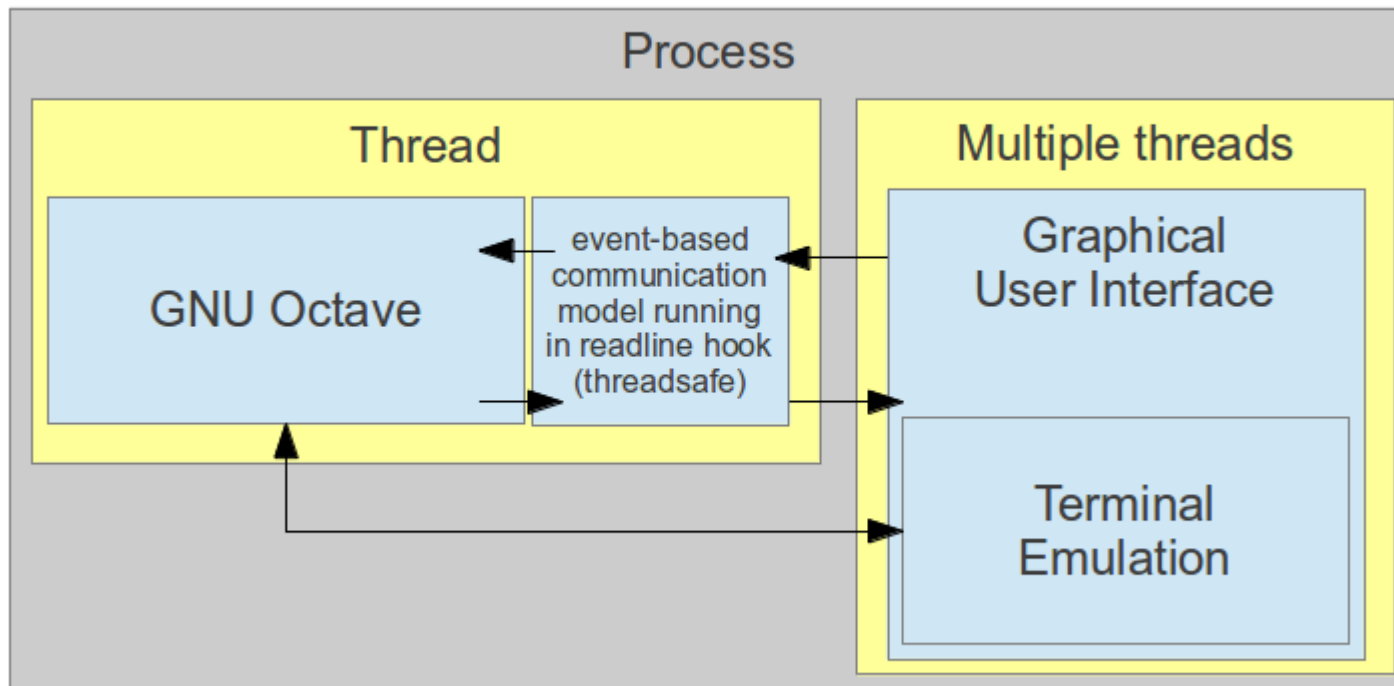
Different ways of running octave

b) GNU Octave running with an external GUI



Different ways of running octave

c) GNU Octave running with GUI



Integrated Editor

- Using QScintilla2 as a powerful editor widget
- QScintilla2 is free software licensed under the GPL
- Offers all the standard comfort of an editor: Marker lines, line numbers, code folding, current line highlighting, syntax highlighting, auto-indent, code-completion
- For further information see:
<http://www.riverbankcomputing.co.uk/software/qscintilla/>

Debugging Interface

- Neatless integration of GNU Octave's debugger. It is possible to set breakpoints at the margin in the editor.
- Stepping through the code can be done with the usual F10/F11/F12 and F5 commands.
- While debugging, it is possible to track the current state of the workspace.

Thank you for your attention!