

# YaST

---

Benjamin Weber <[benji.weber@gmail.com](mailto:benji.weber@gmail.com)>





# Contents

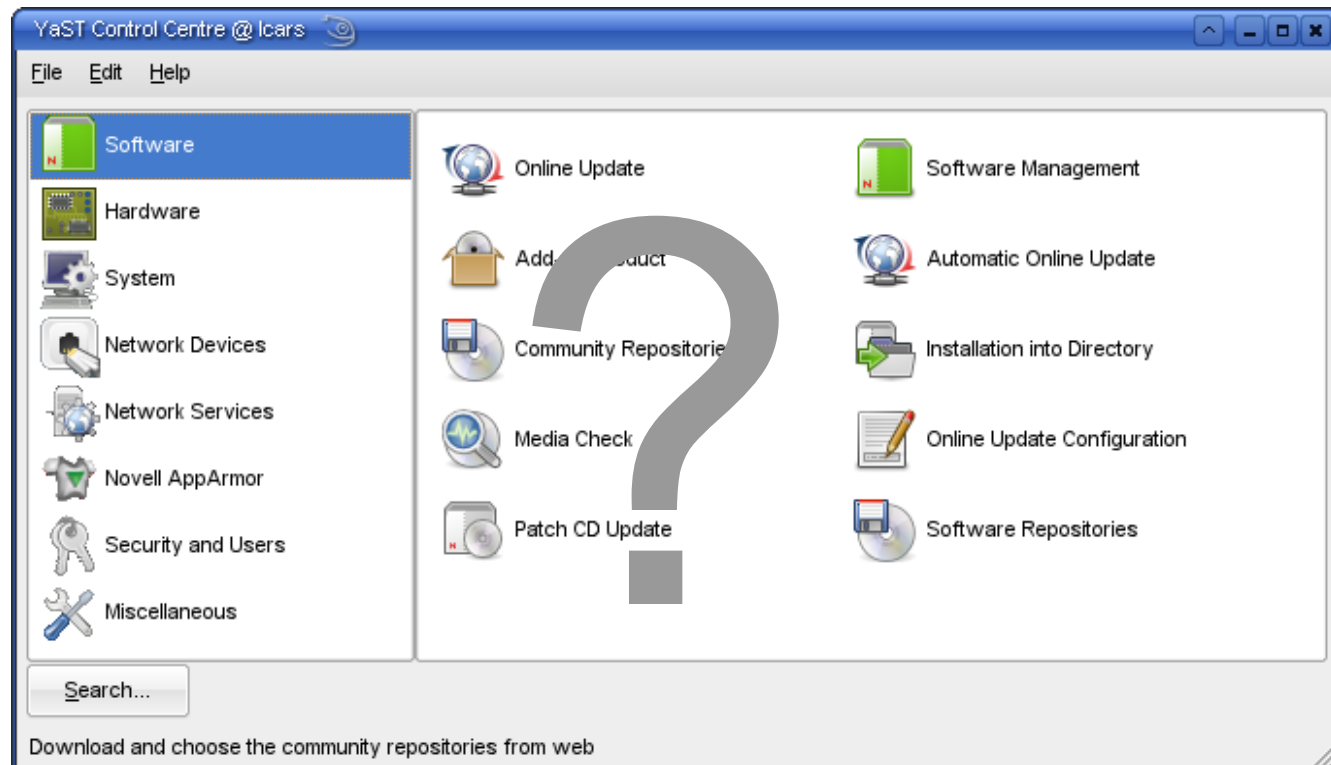
---

- What is YaST
- Details
- Problems & Future of YaST
- Discussion

What is YaST?



# What is YaST?





# What is YaST?

The screenshot shows the YaST2@lcars interface. On the left, there is a search filter set to 'Search' and a search box containing 'amarok'. Below the search box, there are checkboxes for search criteria: Name (checked), Summary (checked), Description (unchecked), RPM "Provides" (unchecked), and RPM "Requires" (unchecked). The search mode is set to 'Contains'. At the bottom left, there is a disk usage table:

Name	Disk Usage	Used	Free	Total
/store/oldhome		81% 189.4 GB	43.4 GB	232.9 GB
/home		80% 562.5 GB	136.1 GB	698.6 GB
/		22% 29.4 GB	102.3 GB	131.7 GB

The main window displays a list of packages with columns for Package, Summary, and Size. The packages listed are:

Package	Summary	Size
<input checked="" type="checkbox"/> amarok	Media Player for KDE	10.2 M
<input checked="" type="checkbox"/> amarok-debuginfo	Debug information for package amarok	44.1 M
<input checked="" type="checkbox"/> amarok-lang	Languages for package amarok	18.0 M
<input checked="" type="checkbox"/> amarok-libvisual	Libvisual Plugin for Amarok	19.0 K
<input checked="" type="checkbox"/> amarok-xine	Xine Output Plugin for Amarok	174.7 K
<input checked="" type="checkbox"/> yauap-gstreamer	yauap gstreamer Backend for Amarok	64.5 K
<input checked="" type="checkbox"/> amarok	Media Player for KDE	20.1 M
<input type="checkbox"/> kxdocker	Kicker Applet that displays what you listen to in Amarok	702.4 K
<input type="checkbox"/> kxdocker-beta	Kicker Applet that displays what you listen to in Amarok	7.3 M
<input type="checkbox"/> kxdocker-gamarok	kxdocker amarok plugin	91.8 K
<input type="checkbox"/> kxdocker-gamarok-debuginfo	Debug information for package kxdocker-gamarok	29.6 K
<input type="checkbox"/> pacpl-amarok	Perl Audio Converter - Amarok extension	2.7 K

A large grey question mark is overlaid on the list. Below the list, there are tabs for Description, Technical Data, Dependencies, Versions, File List, and Change Log. The Description tab is selected, showing the following text:

**amarok** - Media Player for KDE

Amarok is a media player for all kinds of media, supported by aRts, GStreamer or Xine (depending on the packages you install). This includes MP3, Ogg Vorbis, audio CDs and streams. It also supports audio effects of all kinds that are provided by aRts. Playlists can be stored in .m3u or .pls files.

At the bottom of the window, there are buttons for Check, Autocheck (checked), Cancel, and Accept.



# What is YaST?

---

To users...

- Yet another Setup Tool
- System Configuration Tool
- Software Management Tool

... users shouldn't really need to know what YaST is.



# What is YaST?

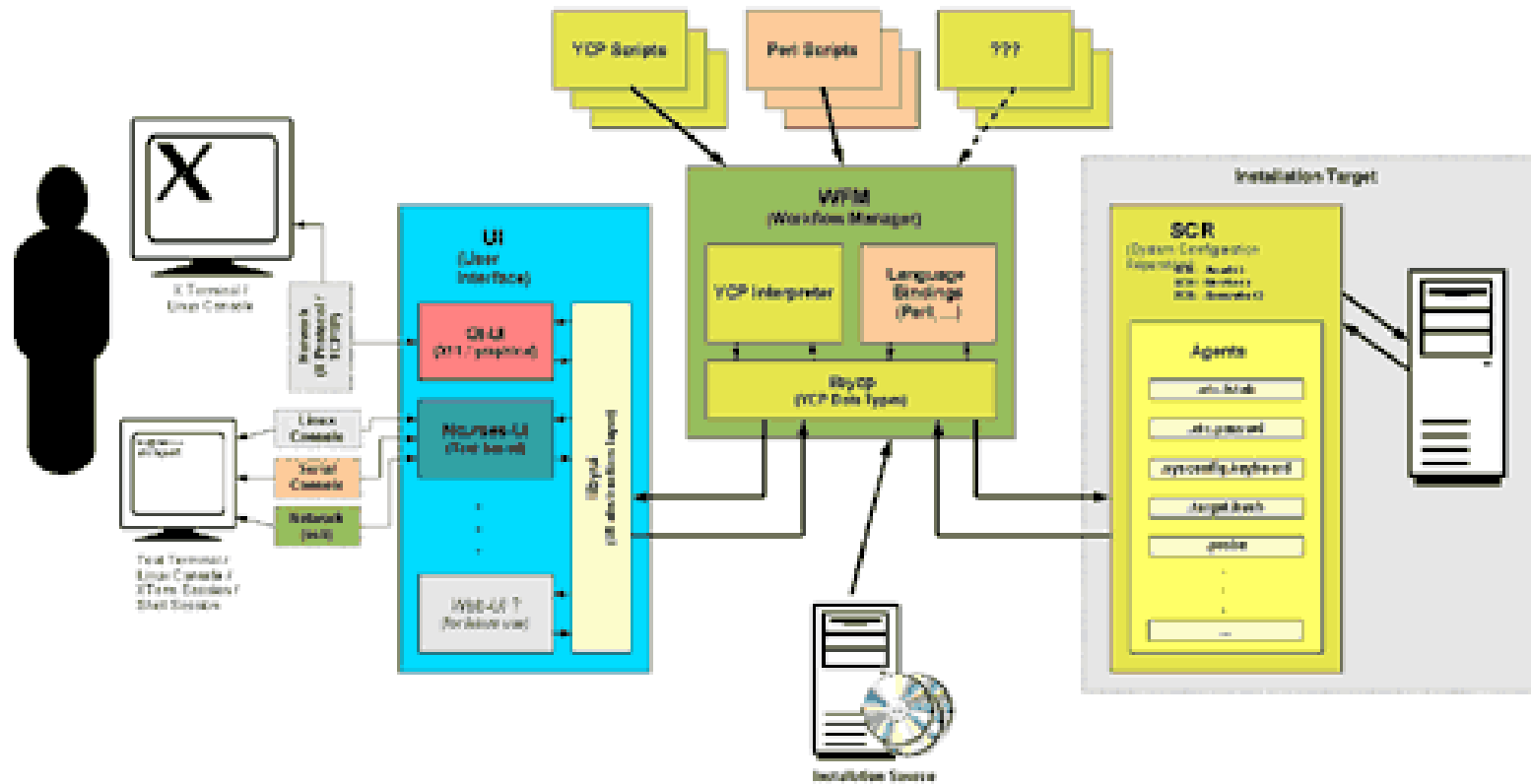
---

To Developers...

- Programming Language
- Runtime Platform
- Component Technology
- Toolkit Independent UI
- System Configuration Repository

# Big Picture

## YaST2 System Architecture



YCP



# YCP Language

---

```
// Hello World
{
    y2milestone("Hello World");
}
```



# YCP Language

---

Unfortunately not Object-Oriented

... But at least we have modules  
(class with only static methods)

.. and typedef

( call structures composed of maps and lists by a name)

.. and pass-by-reference

```
Car::SetSpeed(myCar,50);
```



# YCP Language

---

## Built-in Types:

- any, void, boolean, integer, float, string,
- symbol :       `someSymbol, `someOtherSymbol )
- term :         `someTerm(`one, `two)
- path :         .etc.sysconfig.network
- list:           ["one","two",3]
- map:           \${"one":1, "two",2}
- block         { /\*code here \*/ }
- byteblock   #[123456....

list<string>, map<string,integer>



# ForEach

---

```
list<string> numbers = [ "one","two","three" ];  
foreach ( string str, numbers,  
{  
    y2milestone(str);  
});
```



# Filter

---

```
{
    map<integer,integer> numbers = ${1:2,3:3,4:4};
    map<integer,integer> filtered =
        filter( integer key, integer value, numbers,
            {
                return (key == value);
            });
    y2milestone("new map is %1",filtered);
}

// prints ${3:3,4:4}
```



# YCP/YUI User Interface

---

```
// Graphical Hello World
{
    UI::OpenDialog (
        `Vbox (
            `Label( "Hello World" ),
            `PushButton("&OK")
        )
    );
    UI::UserInput();
    UI::CloseDialog();
}
```



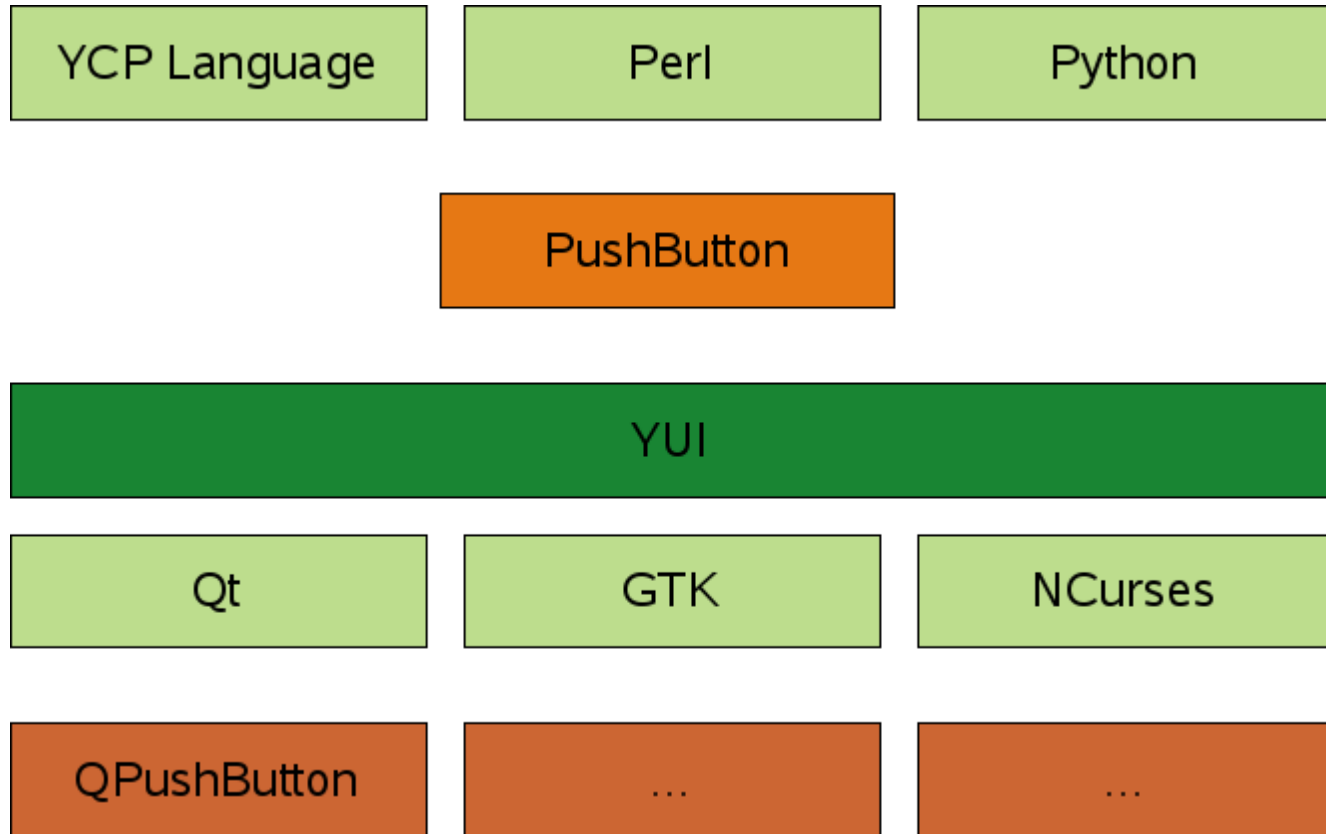
# YCP/YUI User Interface

---

- Toolkit Agnostic (Same code -> GTK, Qt, Ncurses , CLI)
- Widgets implemented for each toolkit.
- Easy to use
- Subset of all means limitations
- Optional Widgets



# YCP/YUI User Interface





# YCP Component Technology

---

- Use External libraries – perl, python, ruby
- Use YCP components from perl, python, ruby
- Even develop UI in perl, python, ruby

SCR



# SCR

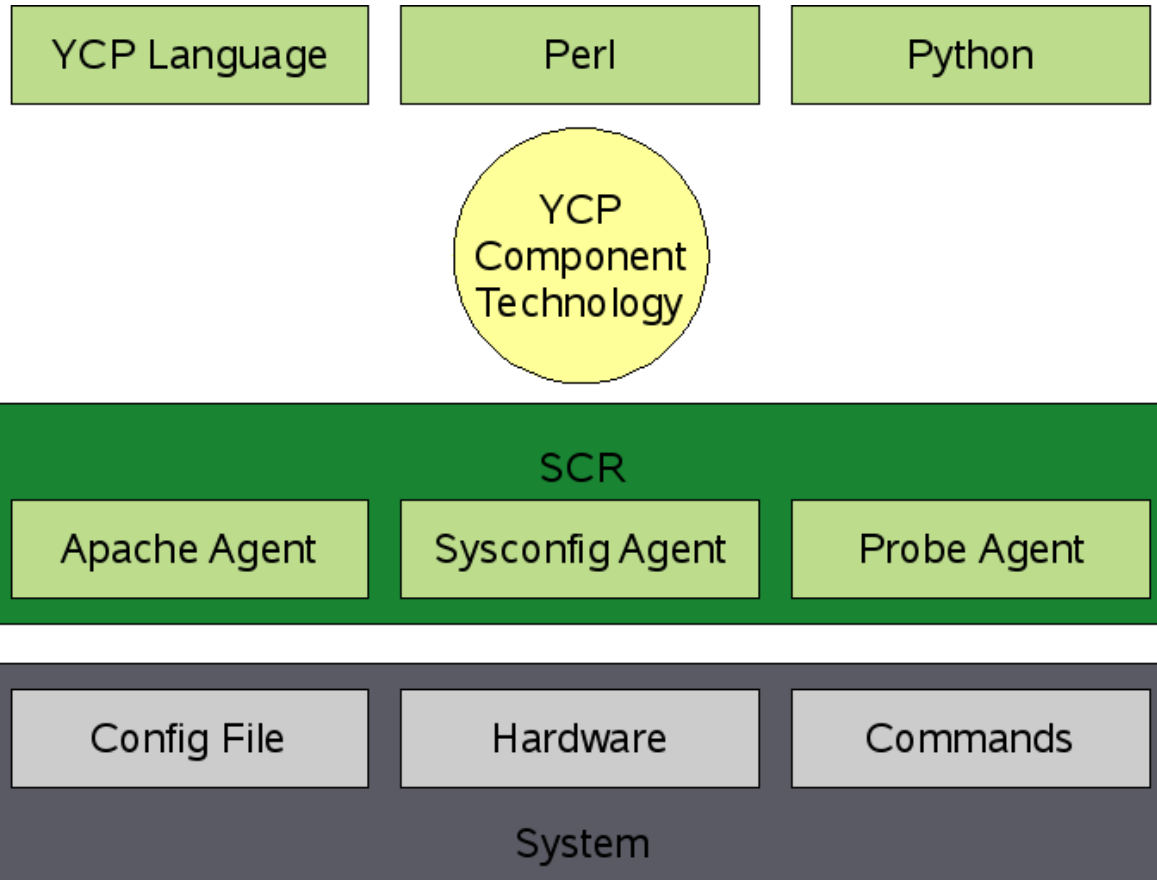
---

## System Configuration Repository

- Read/Write/Execute to arbitrary configuration file formats
- Simple
- Enables Network Transparency
- Agents can be written in any language: c++, perl, bash...



# SCR





# SCR Example

---

```
string xDM = (string)
```

```
    SCR::Read(.sysconfig.displaymanager.DISPLAYMANAGER);
```

```
string arch = (string) SCR::Read(.probe.architecture);
```

```
SCR::Write(.sysconfig.displaymanager.DISPLAYMANAGER_REMOTE_ACCESS, remote_access ? "yes" : "no");
```

```
SCR::Write(topath(sformat(".audio.alsa.cards.%1.channels.%2.mute", card_id, device)), false);
```

```
SCR::Execute(.target.bash, "/usr/bin/test -d ...
```

```
SCR::Execute(.target.mkdir, "/usr/local/wuglug");
```

# Further Resources



# Try things out...

---

## Tools:

- “ycpc -E <file.ycp>” - check a ycp file for errors.
- “ycpc -c <file.ycp>” - compile a ycp file to bytecode.
- “/sbin/YaST2 <file.ybc>” - execute the bytecode.



## Find out more...

---

### Online Resources:

- <http://en.opensuse.org/YaST/Development>
- <http://forgeftp.novell.com/yast/doc/SL10.3/tdg>
- <http://svn.opensuse.org/svn/yast>
- <irc://irc.freenode.net/#yast>
- <mailto:yast-devel@opensuse.org>

# Problems & Future of YaST



# Problems

---

- Old codebase
- Not ported to non-suse distributions
- Difficulty utilising non-yast libraries (except perl...)
- Difficulty utilising YaST features from other applications
- YCP not object oriented – consumable APIs not either
- UI tightly coupled to management – UI running as root



# Solutions

---

These may help...

- ws-management
- DBUS!
- PolicyKit

De-coupling technologies

- Pushing adoption of SCR, YUI as self-contained technologies