

INTO THE new
BREW 2007 CONFERENCE

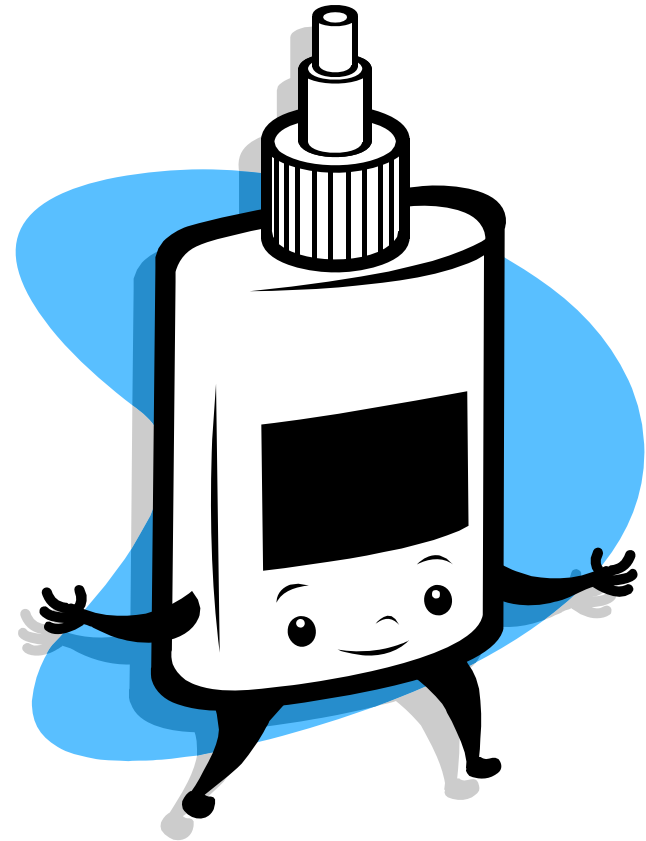
Template Meta
Programming in BREW®

Theodore Witkamp, CTO
RiffWare



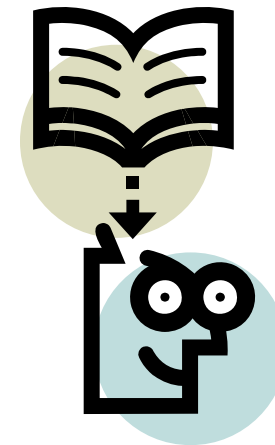
The Problem

- Partial C++ Support
 - Unsupported C++ Libraries
 - Lack of easy Glue
 - Boilerplate code
- ARM ADS 1.2
 - broken Template support
 - no static initializers
 - no C++ namespaces



The Solution

- Full C++ Support
 - Templates
 - > Smart Pointers
 - > Function Objects
 - Static Initializers
 - Namespaces
- ARM RVDS 3.0 + elf2mod
 - C++ 98 Templates





General Overview

C++ Templates

- 1998 C++ Standard
- 2002 C++ Compiler support

BREW[®] Support

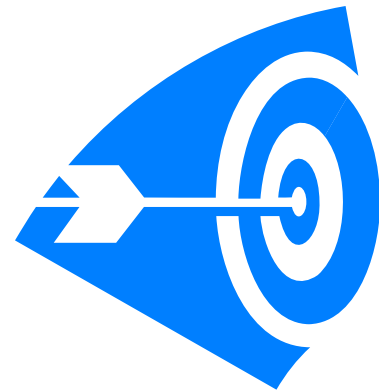
- ARM
 - > ADS 1.2
 - > RVDS 3.0 + elf2mod
- Windows
 - > VC++ 2005

Meta Programming

- Productivity
 - > Smart Pointers
 - > Function Objects
 - > Domain Specific Languages
- Leverage
 - > STL
 - > Other C++ Libraries

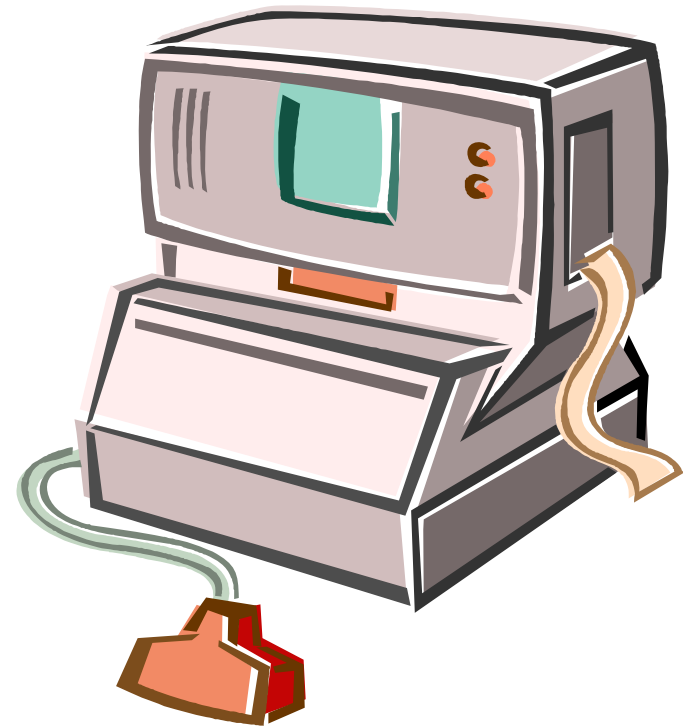
Template Misconceptions

- STL Containers Only
- Parametric Types
 - **Java Generics**
 - C# Generics
- Huge Binaries
 - Post link optimizers can help.
 - > Diablo



Templates: A Compile Time Language

- Turing Complete
- Functional Language
 - No State
- Overhead
 - No Runtime
 - Only Compile Time





What is a Meta Program?

- General
 - Program that Generates a Program
 - Common Example
 - > Parser Generators (antlr,bison,lex...)
- C++
 - Template that Generates a Type
 - Example
 - > `foo<some_type>::type`
 - > `list<foo>::iterator`



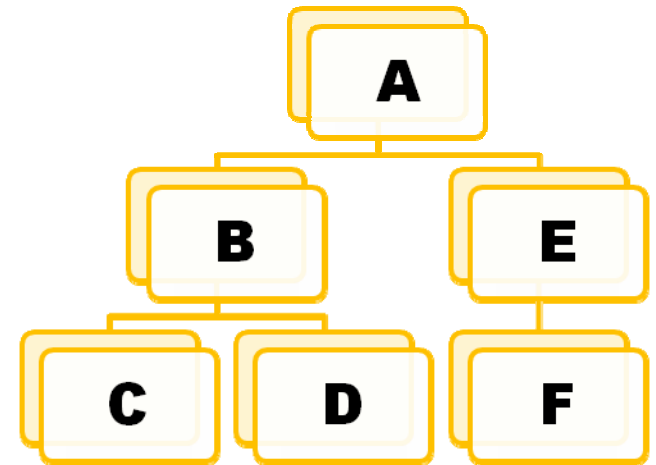
Template Meta Programming

- Common Benefits
 - Smart Pointers
 - Function Objects

- DLS (Domain Specific Languages)
 - Reduce boiler plate code
 - Fix bugs once
 - Improve Productivity

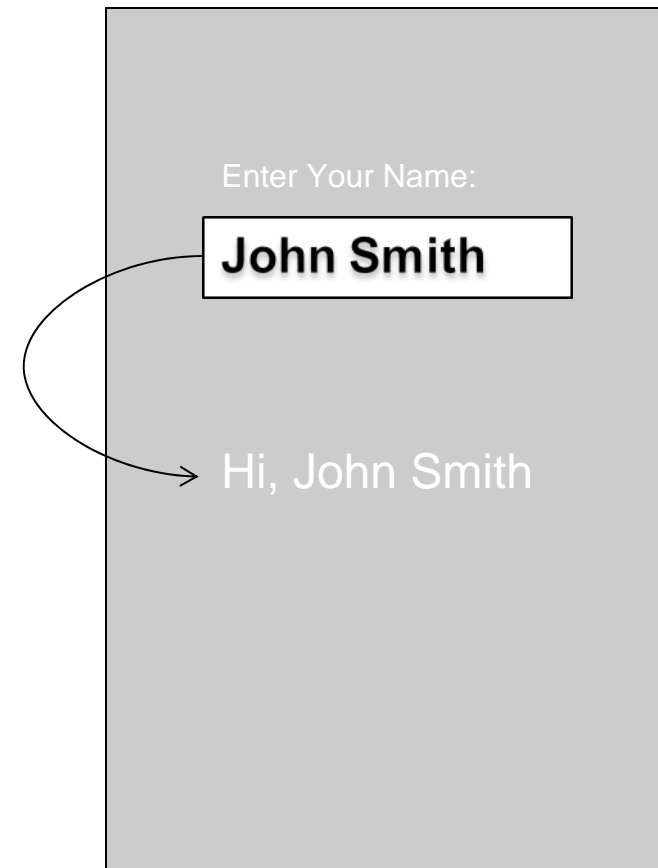
Function Object (Functors)

- Polymorphism
 - No Virtual Dispatch
 - Type Safe
- Sub Typing Not Needed
- Code Size
 - Proportional to number of Types
 - No Worse Than doing by Hand



Function Object Simple Example

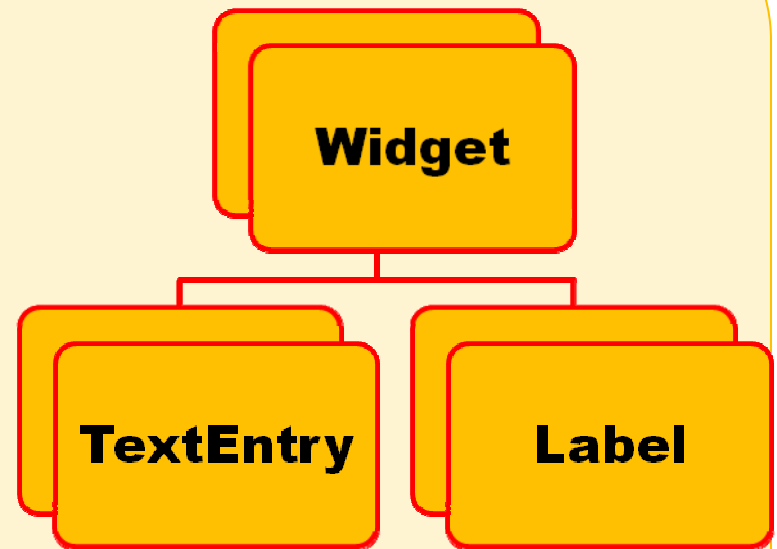
- Echo the Text Entry
- Goals
 - Reuse the Code
 - > Text Entry
 - > Text Label
 - No Sub Types
 - No Listener Objects
 - Little Typing



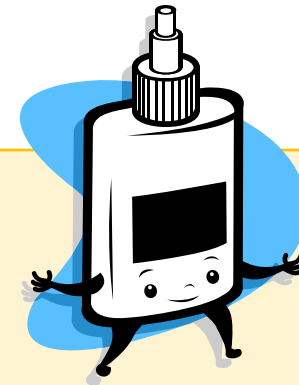
Example Class Hierarchy

```
// Text Entry Widget
class TextEntry: public Widget
{
public:
    String getString();
    void callOnUpdate(function<void(>
func);
};

// Label Widget
class Label : public Widget
{
public:
    void setString(String string);
};
```



The Glue Code



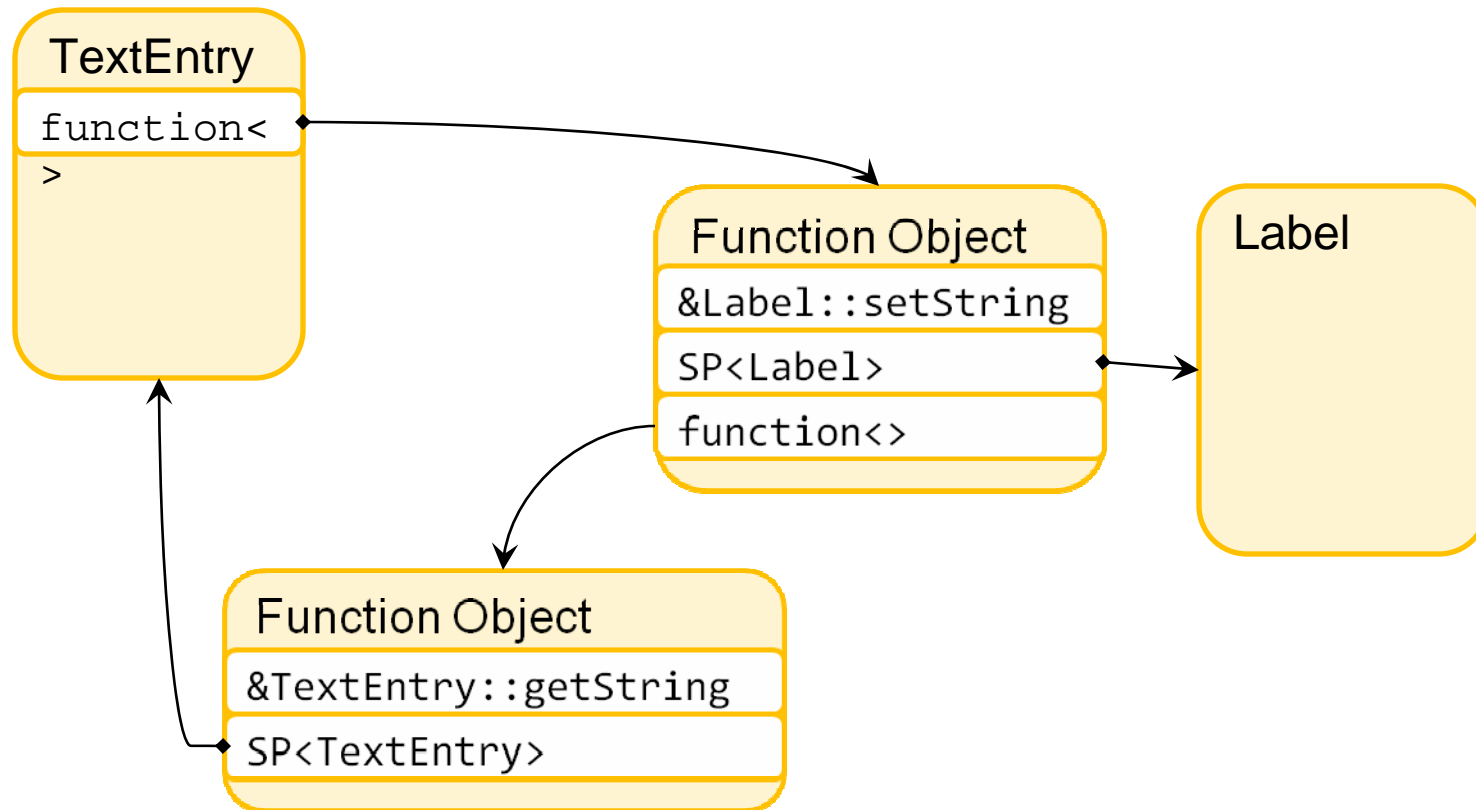
```
// Create the Object Instances
SP<TextEntry>  textEntry = new TextEntry();
SP<Label>      label      = new Label();

// Create The Function Object to get The String
function<String()> get_string(&TextEntry::getString, textEntry);

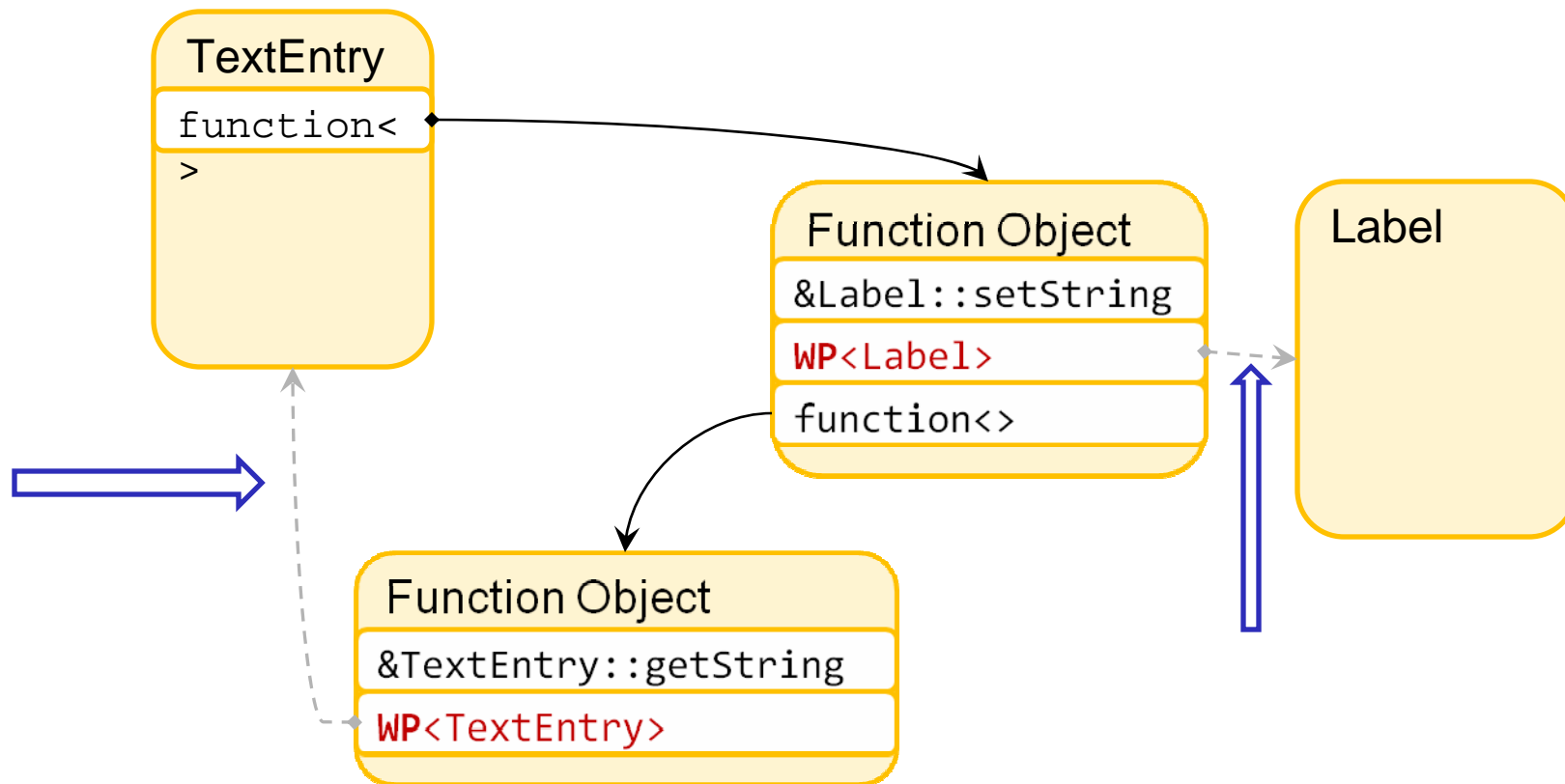
// Create The Function Object to update the Label
function<void()> update_label(&Label::setString, label, get_string)

textEntry->callOnUpdate(update_label);
```

Oops! Cyclic References



Breaking the Loop



Revised Glue Code



```
// Create the Object Instances
SP<TextEntry>  textEntry = new TextEntry();
SP<Label>      label      = new Label();

// Create The Function Object to get The String
function<String()>
get_string(&TextEntry::getString,make_wp(textEntry));

// Create The Function Object to update the Label
function<void()>
update_label(&Label::setString,make_wp(label,get_string));

textEntry->callOnUpdate(update_label);
```

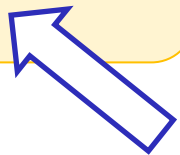
make_wp Template Code

- Saves Typing
- Reduces references to types
- No runtime cost
- Code is inline
- It's Free!

```
template<typename T>
inline WP<T> make_wp(SP<T> sptr)
{
    return WP<T>(sptr);
}
```

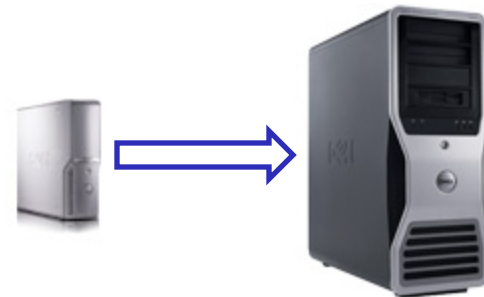
```
// without make_wp
foo(WP<TextEntry>(some_ptr));

// with make_wp
foo(make_wp(some_ptr));
```



Templates on BREW[®]

- Windows Template Support
 - We Use VC++ 2005 Internally
- ARM Template Support
 - We Use RVDS 3.0
- Code Costs For Function Objects
 - Proportional to Types Used
 - ~10-15k of Thumb code
- Compile Time Increased
 - Buy a real workstation!





ARM Template Support

ARM ADS 1.2

- lots of bugs
 - Partial specialization
 - smart pointers work
- C++ namespaces
 - NOT supported
- ROPI C++
 - Supported!

ARM RVDS 3.0

- EDG C++ Frontend
 - **E**dison **D**esign **G**roup
- Full C++ support
 - Templates
 - Namespaces
- ROPI C++
 - Not Supported!
- elf2mod
 - Required!



Getting RVDS 3.0 Up And Running

- Change Build
 - NO Position Indendent Code (ROPI)
 - `armlink --reloc`
- Implement the RVDS Semi Hosting
 - malloc/free and other runtime requirements
 - call the runtime functions
 - > `__rt_lib_init(0,0);`
 - > `__rt_lib_shutdown();`
- elf2mod workaround for RVDS 3.0 SP1
 - `armlink -ro-base 0`



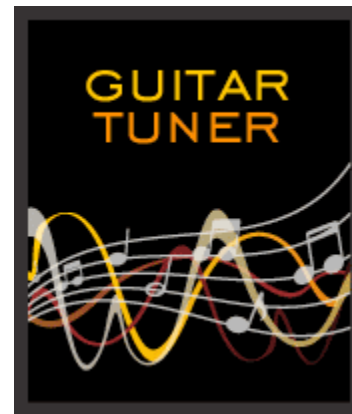
Summary

- RVDS 3.0 + elf2mod
 - Provides a Modern C++ Language Platform
 - Well Supported C++ Frontend

- Template Meta Programming
 - Function Objects
 - > Great Glue for Components
 - Saves Developer Time
 - No Need for external Tools
 - Lots of Options (DSL..)

Results

- Weeks Not Months



Application
Stitching Time

3 weeks



1 week