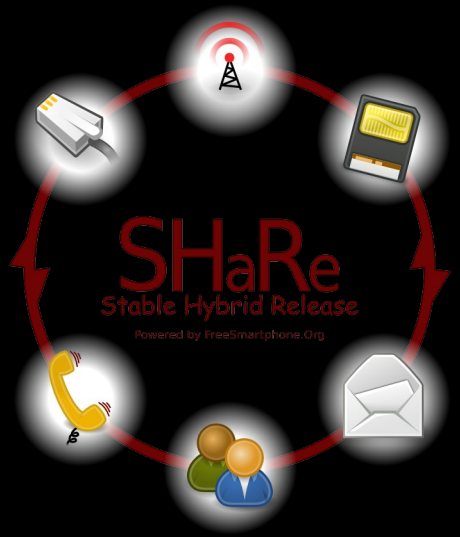
Ophonekitd/Vala

A Vala rewrite of Ophonekitd

FSO|SHR '09
Users & Developers Convention



Ophonekitd/Vala – Summary

- Why Vala ?
- Libmodulo – Design principles
- Libmodulo – Abstractions
- Architecture
- Current state
- Conclusion

Ophonekitd/Vala – Why Vala ? Object oriented features

- Namespaces
  - File organisation in developers' hands
- Full abstraction and inheritance support
- Properties
- Signals
- Delegates
- Closures
- Ownership transfer
- Annotations

Ophonekitd/Vala – Why Vala ? Generated C backend

- Profile-based
  - Multiple backends (currently GLib, Posix)
  - Vala features supported depending on backend
- GLib backend
  - GObject and GValue integration
- Fully compatible with C
  - Implement Vala interfaces in C
  - Call Vala code from C
  - VAPI ease bindings to C libraries

Ophonekitd/Vala – Why Vala ? Integrated DBus support

- Binding to DBus through annotations
- “Static” and “dynamic” support
  - Dynamic provides untyped calls (still dbus-glib)
  - Static provides for generated marshalling code
- Transparent support for:
  - Parameter marshalling
  - Enumerations marshalled as strings
  - GError marshalling
  - ObjectPath/Proxy substitution (WIP)

Ophonekitd/Vala – Why Vala ? Asynchronous programming (WIP)

- Will support asynchronous calls
  - Method signature declares that it '`yields`'
  - Call from synchronous context with `method.begin()`
  - Expected synchronous-like call syntax ('`yield`' ?)
  - Callback-based alternative still available
- Broad library support
  - IO primitives (GNIO), DBus (integrated in Vala), …
- Server-side DBus support (due 2009-06-29)
  - Just like any other asynchronous methods

Libmodulo – Design principles Dependency inversion

- Traditional programming
  - High-level modules depend on low-level modules
- Inverted flow of control
  - High-level modules and low-level modules depend on a shared abstraction
- “Dependency inversion”
  - Every object collaborate with abstractions

Libmodulo – Design principles Dependency injection

- Traditional programming
  - An object instantiates and uses its dependencies
- Inverted flow of control
  - Dependencies are instantiated outside the object and injected in the object
- “Dependency injection”
  - Every object are given their collaborators

Libmodulo – Design principles Component-oriented programming

- Components
  - As the unit of deployment
  - Provide services for other components
  - Use services from other components
- Container
  - Instantiate the components
  - Inject components one in another as required

Libmodulo – Design principles Component-oriented programming

- System designer
  - Defines abstractions for the services
- Component developers
  - Implement components independently
- System assembler
  - Selects and assembles components

Libmodulo – APIs Overview

- Components
  - Have lifecycle
  - Can log about their state
  - Can be configured
  - Use and provide services
  - Have lifestyle
- Container
  - Instantiate components
  - Wire and configure them

Libmodulo – APIs
Component instantiation

- Multiple component factories
  - GObject component factory
  - Remote DBus component factory
- Expected component factories
  - Wrapped Python component factory
  - ...

Libmodulo – APIs
Lifecycle management

- Initializable
  - `void initialize()`
- Releasable
  - `void release()`
- Startable
  - `void start()`
  - `void stop()`
- Suspendable
  - `void suspend()`
  - `void resume()`

Libmodulo – APIs
Logging

- Logging
  - `void set_logger(Logger logger)`
- Logger
  - `void error(Object source, string message)`
  - `void info(Object source, string message)`
  - `void warning(Object source, string message)`
  - `void critical(Object source, string message)`
  - `void debug(Object source, string message)`

```
if (Logging.enabled(logger, LogLevel.DEBUG)) {
    logger.debug(this, "A log message");
}
```

Libmodulo – APIs
Configuration

- Configurable
  - `void configure(Configuration configuration)`
- Configuration
  - `string get_name()`
  - `string get_string(string name)`
  - `Configuration? get_child(string name)`
  - `Configuration[] get_children(string? name)`

Libmodulo – APIs
Servicing

- Serviceable
  - `void service(ServiceManager service_manager)`
- ServiceManager
  - `Bool has(string key)`
  - `Object lookup(string key)`

Libmodulo – APIs
Component metadata

- Using Vala annotations
  - On the provider side

```
[Modulo (provides="Foo.FooService")]
public class Bar : FooService, Logging, Serviceable, Object {
```

  - On the user side

```
[Modulo (requires="foo:Foo.FooService")]
public class Baz : Logging, Serviceable, Initializable, Object {
```

  - Export the service through DBus

```
[Modulo (…, dbus_export_path="/org/shr/Foo")]
public class Ba
```

Libmodulo – APIs
Container configuration

- Using Vala annotations
  - On the provider side

```
<component name="bar"
      factory="Modulo.GObjectFactory" type="FooServer.Bar">
      <configuration/>
</component>
```

  - On the user side

```
<component name="foo" factory="Modulo.DBusFactory"
      bus-name="org.shr.ModuloTest" object-path="/org/shr/Foo"/>

<component name="baz"
      factory="Modulo.GObjectFactory" type="FooClient.Baz">
      <service name="foo" component="foo"/>
      <configuration/>
</component>
```

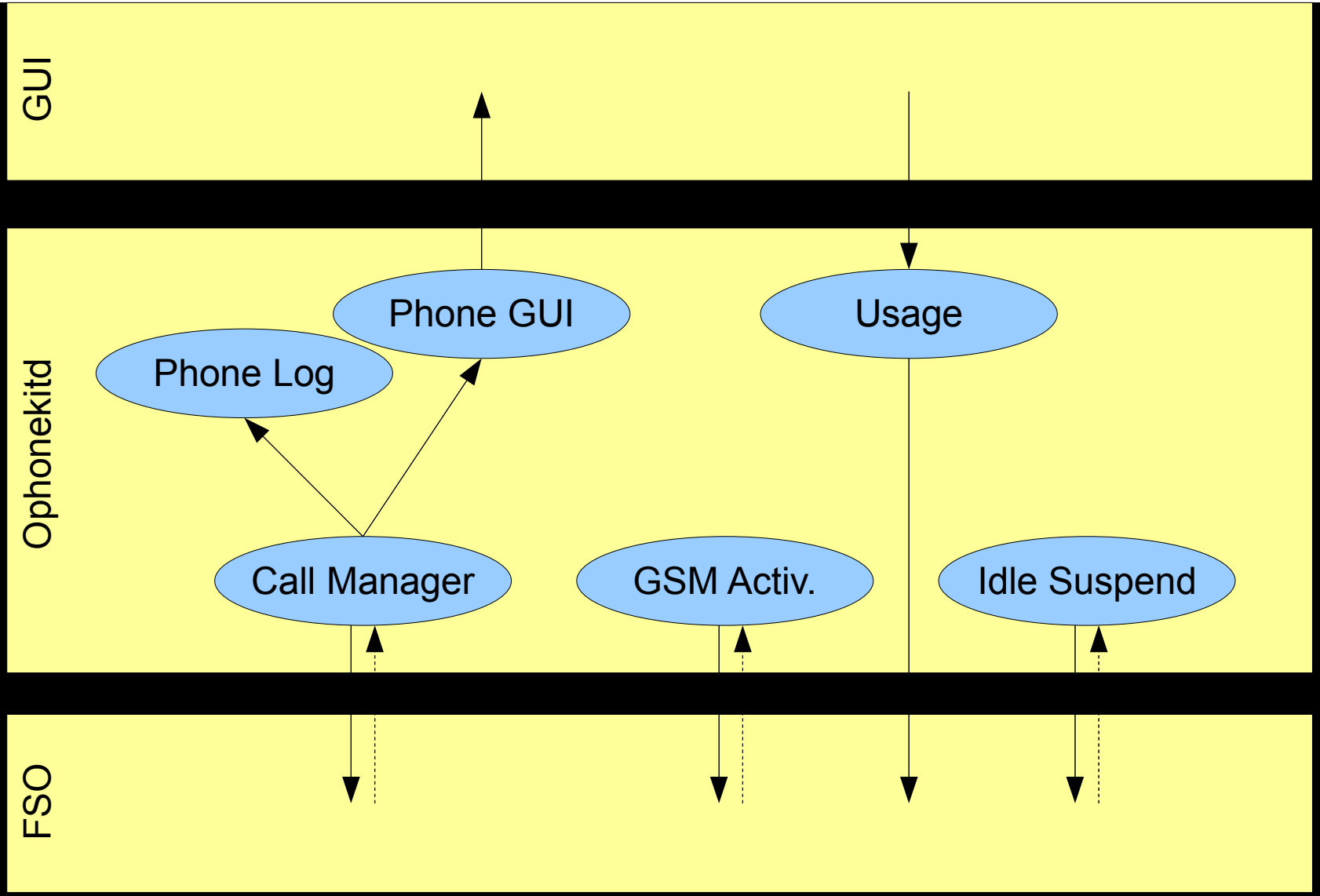
Libmodulo – APIs
Lifestyle management (NYI)

- Currently used lifecycle
  - All components are singletons
- Expected lifecycles
  - Instantiation directives
    - Per request instances
    - Per thread instances
  - Threading directives
    - Thread safe
    - Thread pooled

Ophonekitd – Architecture Components

- Concerns of ophonekitd
  - GSM activation
  - Idle suspend
  - Call management
  - Phone log
  - Phone GUI
  - Usage service

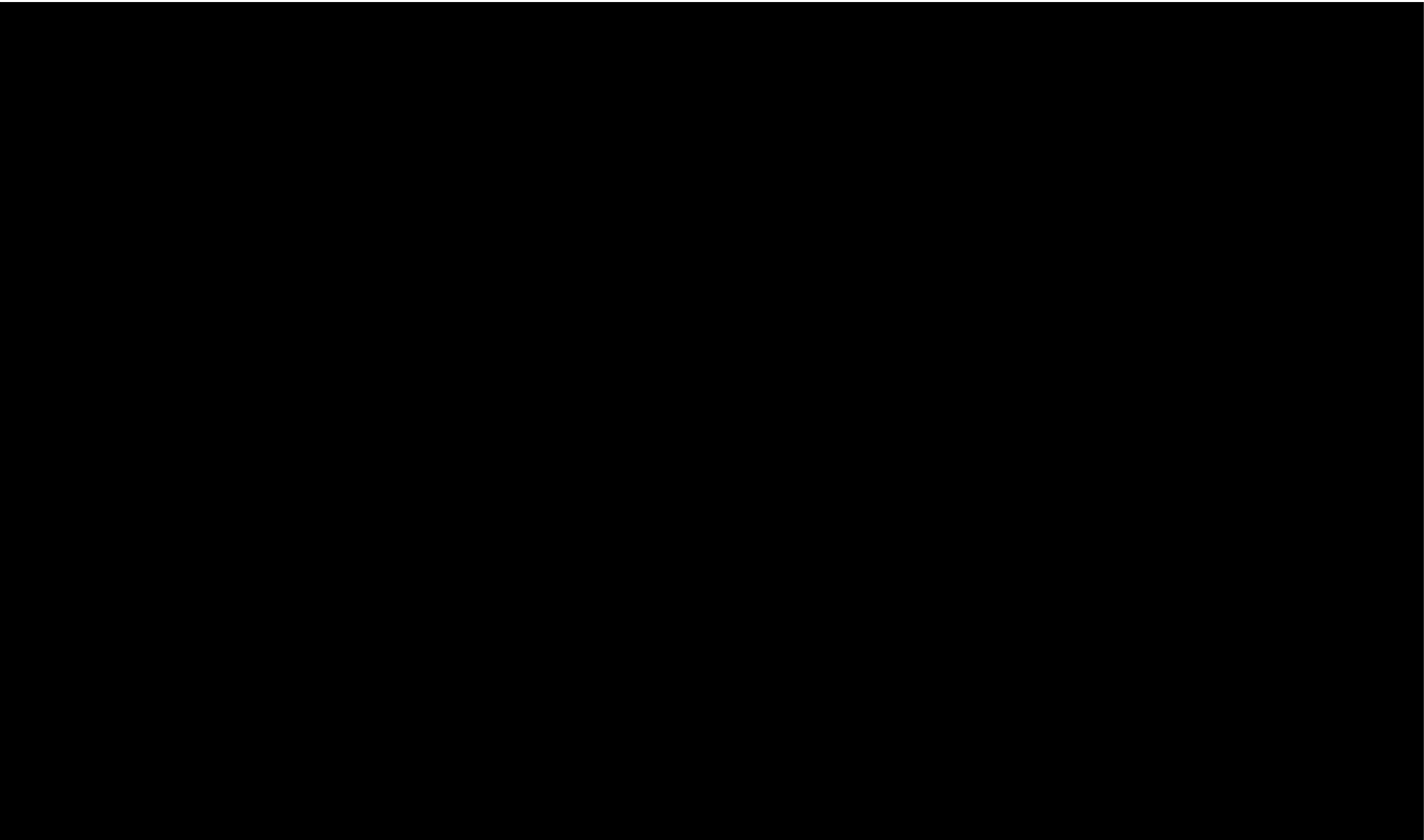
Ophonekitd/Vala – Architecture Components



Ophonekitd/Vala – Current state

- Libmodulo
  - Implement Lifestyle
  - Extend to support more dynamic scenarios
- Ophonekitd/Vala
  - Substitute CallManager by FreeSmartphone.Phone
  - Split PhoneGUI in functional parts
  - Extract DBus service definitions

Ophonekitd/Vala – Conclusion



Ophonekitd/Vala – Resources

- Libfso-glib:
  http://git.freesmartphone.org/?p=libfso-glib.git
- Libmodulo:
  http://git.shr-project.org/git/?p=libmodulo.git
- Ophonekitd/Vala:
  http://git.shr-project.org/git/?p=ophonekitd.git
- Inversion Of Control:
  http://en.wikipedia.org/wiki/Inversion_of_Control