ComponentModelFunctionPointers

Proposed Component Model written in C

A proof-of-concept implementation can be found here: http://issues.apache.org/jira/browse/HARMONY-5

Source Files

- `vm/core/comp_model/VmComponent.h` – defines `ComponentInfo`, `DependencyInfo` and the functions every component has to provide.
- `vm/core/comp_model/ComponentModel.[ch]` – The central files of the component model
- `vm/core/comp_model/interfaces/TestComponent1.h` – The interface of `TestComponent1`
- `vm/core/comp_model/interfaces/TestComponent2.h` – The interface of `TestComponent2`
- `vm/components/TestComponent1Impl/TestComponent1Impl.c` – An implementation of `TestComponent1`
- `vm/components/TestComponent2Impl/TestComponent2Impl.c` – An implementation of `TestComponent2`
- `vm/test/comp_model/TestComponentModel.c` – A simple test case for the component model.

Description

All components are packed in a shared library and are loaded by the component model with dlopen. The component model will load all the components it can find in a certain set of directories which has to be passed to the function "componentModelInitialize". The components have to provide the following functions:

```
/** Get the ComponentInfo object for this component.
 */
ComponentInfo *getComponentInfo();

/** Get the number of dependencies of this component.
 */
int getNumDependencies();

/** Get a dependency of this component.
 */
DependencyInfo *getDependency(int i);

/** Initialize the component (will not be called before all dependencies have been set).
 */
void initialize();
```

`ComponentInfo` and `DependencyInfo` are structs which contain information about the component and it's dependencies. When the component model loads the components it first gets the `ComponentInfo` entries for all components and resolves the function pointers for the three functions mentioned above. In a second step it resolves the dependencies of all components and injects the dependent components with the setter function from the `DependencyInfo` entry. After that it calls `initialize` for all components.

A component which depends on another component gets the function pointer table of this component set by the component model. For example, `TestComponent1` gets a pointer to a `TestComponent2` and can then call the functions defined there.

Building and running the test case

It can be built with

```
libtoolize
aclocal
autoconf
automake
./configure
make
```

The test case can be started from the vm-directory with the command

```
test/comp_model/TestComponentModel
```