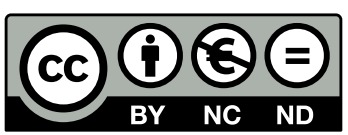
Component technology in an embedded system

David Polberger

Master's thesis defense
January 27, 2010



Background

- I started an internship at Sony Ericsson in 2005.
- I worked on the company's component technology.
- My goal was to make it possible to trace invocations.

Objectives

- Make sense of components, thereby bringing order to chaos.
- Seek insight by creating an object model and sketching a component model.
- Explain important technical concepts in software engineering.

Objectives, continued

- Investigate component models in use in industry.
- Place Sony Ericsson's component technology in a historical and technical context.
- Enable execution tracing at Sony Ericsson.

What are components?

- Components encapsulate discrete functionality, and may be combined to form complete solutions.
- Components may be bought and sold on a market.
- "Buy, don't build."

Contemporary components

- A component is a stand-alone entity which can be independently deployed.
- A component is encapsulated, and is only accessed using well-specified interfaces.
- A component is generic, and is customized to fit different domains.

Contemporary components, continued

- A component is a container of classes, which when instantiated form objects.
- A component is written for a specific *component model*.
- Component models dictate the standards components play by.

Components and other forms of reuse

- Software reuses functionality in a variety of ways:
  - ...from the operating system through system calls.
  - ...from libraries through procedural or object invocations.
  - ...from the Internet through web services.
  - ...through pipes.
- Software reuse per se cannot be credited to components.
- Components bring increased rigor through standards.

Terminology

- The “component” word is laden with multiple meanings.
- In some communities, a “component” is a mere class.
- In others, a “component” is manipulated visually in a designer.
- Many synonyms: *module*, *package*, *bundle*, *assembly*, *server*...

First-generation component models

- Such components are manipulated in visual designers.
- Custom code is added to handle events.
- They provide rapid application development for the desktop.
- Early versions of Microsoft's Visual Basic and Borland's Delphi are prominent examples.

Second-generation component models

- Far more ambitious.
- No longer tied to a single environment.
- Features for the enterprise.
- Examples include Microsoft's COM and OMG's CORBA.

Second-generation component models, new features

- Language-agnosticism.
- Location-transparent invocations.
- Declaratively enabled services for use in the enterprise.

Second-generation component models, complexity

- A host of facets are standardized:
  - Classes, objects and interfaces.
  - Memory management.
  - Error handling.
  - Network protocols.
  - Even software components!
- As a result, very complex.

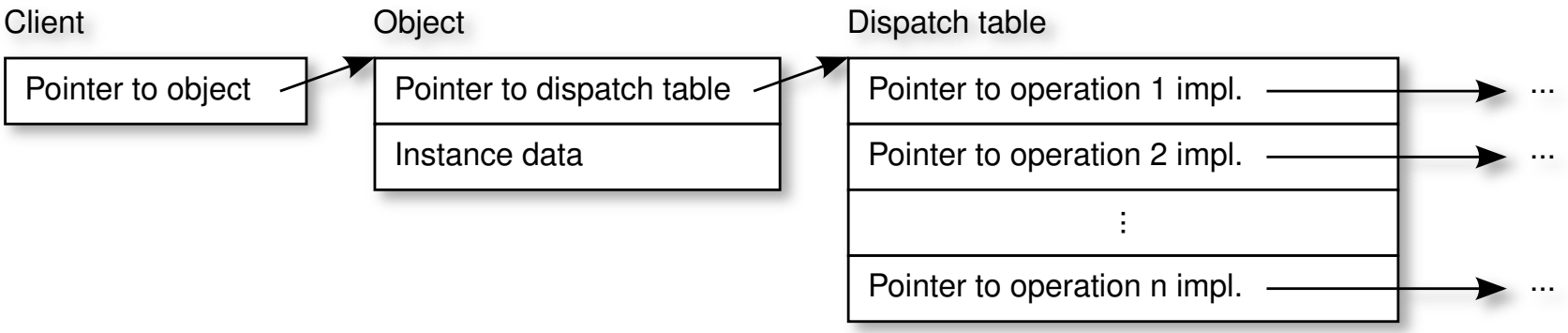
Third-generation component models

- Based on virtual machines that solve many of the problems tackled by second-generation models.
- Only one virtual machine is targeted, instead of many disparate native machines. Simplicity follows.
- Such models thus focus solely on software components.
- Examples include OSGi for Java and .NET assemblies.

Interfaces

- The implementation of a component is hidden.
- This is realized through the use of interfaces.
- Interfaces require *late binding*, which entails locating the implementation at runtime.
- Late binding is typically realized using a runtime data structure, a *dispatch table*.

Dispatch tables



Microsoft's Component Object Model

- An object and component model primarily on Windows.
- Dispatch tables play a major role in COM.
- Interfaces can be described in an *interface description language* (IDL).
- An *IDL compiler* is used to generate code from IDL files.

Sony Ericsson's Ericsson Component Model Extended

- Used for internal development.
- Heavily inspired by COM.
- A custom IDL compiler is used with a custom IDL dialect.
- New features include support for transparent inter-process communication, a Java binding and execution tracing.
- Only an object model for now.

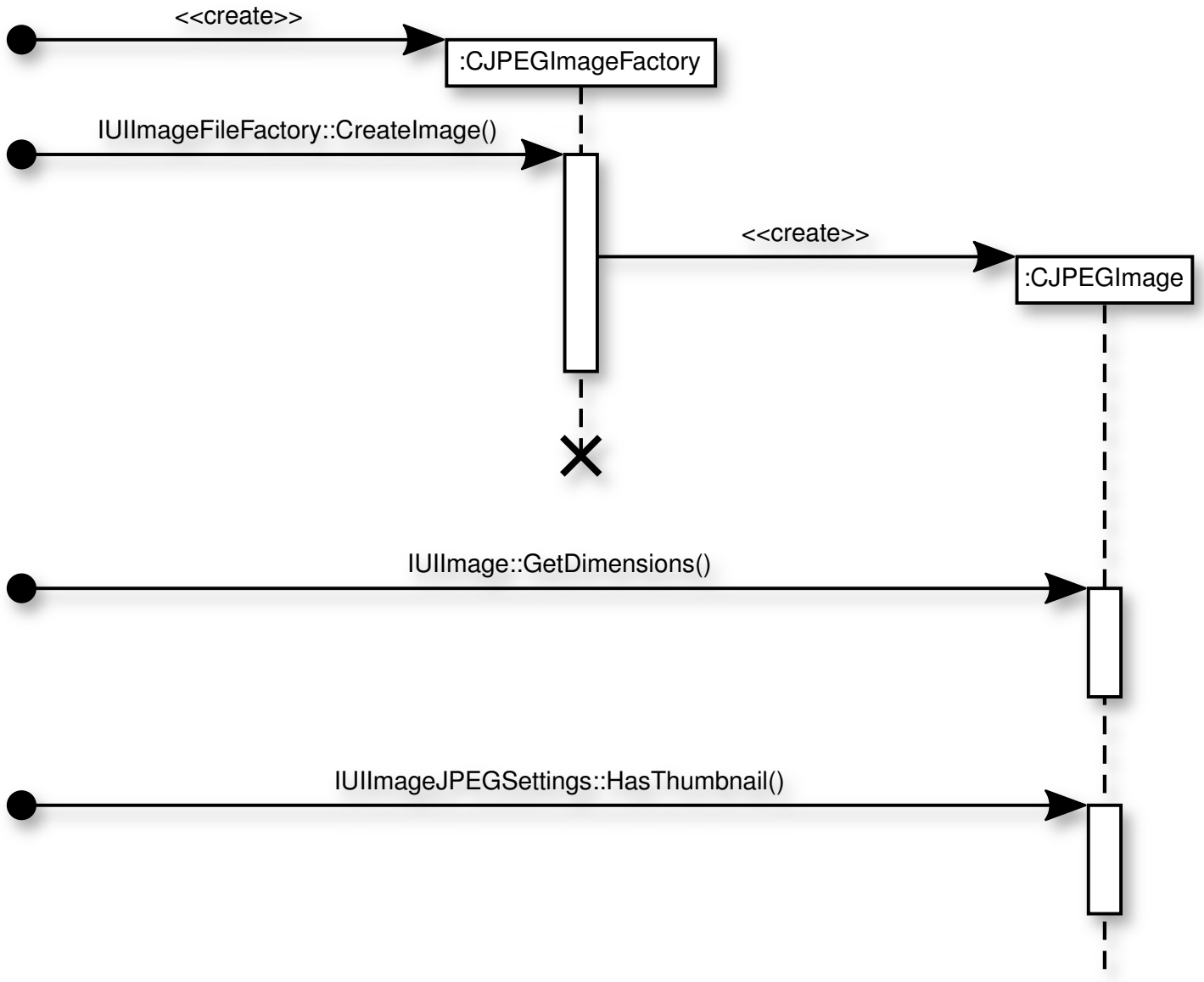
Execution tracing at Sony Ericsson

- Execution tracing was added to ECMX as part of this work.
- Invocations to objects are *intercepted* and logged.
- Trace files may be converted to UML sequence diagrams.

A sample trace file

```
C98487__2c434f7434124CJPEGImageFactory__
E98487__2c434f7434200CJPEGImageFactory_IUIImageFileFactory_CreateImage
C98487__2c56cc7c34264CJPEGImage__
L98487__2c434f7434404CJPEGImageFactory_IUIImageFileFactory_CreateImage
D98487__2c434f7434436CJPEGImageFactory__
E98487__2c56cc7c34484CJPEGImage_IUIImage_GetDimensions
L98487__2c56cc7c34528CJPEGImage_IUIImage_GetDimensions
E98487__2c56cc7c34576CJPEGImage_IUIImageJPEGSettings_HasThumbnail
L98487__2c56cc7c34668CJPEGImage_IUIImageJPEGSettings_HasThumbnail
```

A sample sequence diagram



Execution tracing at Sony Ericsson, implementation

- The IDL compiler has been modified to make the generated dispatch tables call generated wrapper functions.
- The wrapper functions log the invocation, call the original operation and log the commencement of the call.
- Tracing is enabled on a per-class basis.

Evaluating the ECMX trace feature

- Works well in practice.
- Makes it easy to debug distributed functionality.

Evaluating the ECMX trace feature, continued

- Hampered by slow build times.
- A COM+-esque solution would be preferable.
- UML sequence diagrams not ideal.

Summary

- Components encapsulate discrete functionality, and are combined to form larger solutions.
- Second-generation component models standardize everything from objects to network protocols.
- Platforms used by third-generation component models subsume many of the technologies standardized by second-generation models, thus simplifying the technology.

Summary, continued

- Components are only accessed indirectly through interfaces.
- This indirection makes it possible to intercept calls.
- This thesis has made use of interception to implement execution tracing at Sony Ericsson.

Evaluating component technology

- A hodgepodge of technologies, ostensibly under one umbrella (named *Component-Based Software Engineering*, or CBSE).
- The component terminology is not widely used.
- The component technology, on the other hand, is widely used.

More on the Web

http://www.polberger.se/components/