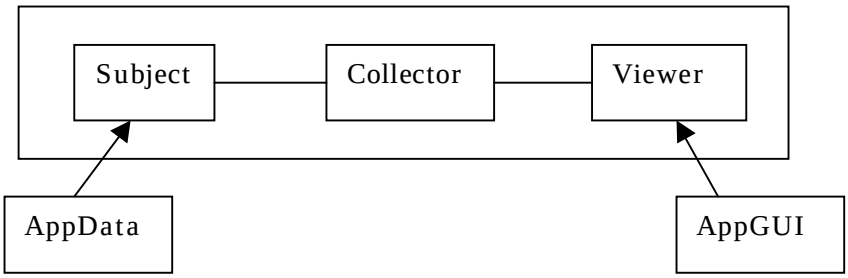
ICSE98 Joint Workshop #05 Position Paper
Title: From Class Libraries to Component based development.
Name: Takeshi Inoue
Organization: Yokogawa Electric Corporation

Background:
We are thinking about dveloping a reusable library for our domain's system construction. Ideally, the system should be constructed in a fully component based fashion but we do not think we can construct a system by only composing software components in current technical levels. It is true that the system should have several spots that accept software components, but when we see the system as a whole, the component-surrounding frame must be developed from scratch or we must use some "class library"-based frameworks. In order software objects to be flexible or fully customizable, class libraries or white-box style reuse seems still advantageous.

Discussion:
We prepared a very simple MVC style component system and constructed a simple application on it as an experimental program. In the component system, 3 classes, i.e. viewer, collector, and subject are prepared as Java based component classes. These 3 classes have no knowledge of application's data, or application dependent information. An application developer must prepare data classes and GUI classes. A data object will tell its data and data accessing methods to Subject component. A GUI object will tell its interests to the Viewer class and at the same time, the GUI object asks the Viewer to call a method on GUI when the data is updated. The Collector component manages matchings between Subject and Viewer such as the mapping of Viewer and Subject, data acquiring intervals, etc. Currently, application developers must write a script-like Java codes to create and use component instances. These features are described in a diagram below.



These 3 components will eventually be integrated to one interface component where a developer asks every component management operation to it.

This component model will also be enhanced in 3 kinds of directions.
+inter-component communication logic, i.e., data pushing(even driven), pulling(scan-bsed).
+to be able to cope with distributed object system using CORBA, DCOM, etc.
+dynamic plugging in and out of data and GUI object to the component.
Before implementing these items, we have to think about how this model should be used in real systems, how this is flexible to cope with real systems' requirements, and how this should become useful. I am trying to evaluate it in terms of black-box and white-box style reuse.

This simple MVC model could easily be constructed by preparing white-box style class library where application developers implement data and GUI classes as subclasses of the Subject and the Viewer respectively. We avoided this by using black-box style component to overcome white-box style reuse's weak points, i.e., robustness, maintainability, etc.

In order software reuse to be widely accepted and component market to be popular in our business market, ideally, black box style reuse should be dominated. However, when we think about current technology and domain research levels, white-box style class libraries and frameworks will remain for a while or forever because of its flexibility.

There are trade-offs between black-box approach and white-box approach. We have to select

one when developing reusable libraries and there must be some determining point to select either of them. This point will fluctuate according to not only technological evolution status but also comprehension and application experiences in target domains. The point also depends on factors such as, domains, languages, objects' characteristics, objects' flexibility and stability, etc. These factors must be studied in more details.

Technically, one of the key points to shift to black-box style reuse is how black-box component can have white-box's superior points, i.e., extendibility and flexibility. On the other hand, if component users need not require limitless extendibility or flexibility for components, this will also solve the key; this means all components are well constructed and prepared as "component pattern" and all you have to do is just to choose one pattern from the pattern library. In order to establish this, we, again, have to analyze and categorize target domain.

I want to discuss about turning points from white-box class libraries to black-box component as well as all related component based software development issues such as reuse, business, standardization, etc.