Tailorable Software Architectures in the Accelerator Control System Environment

Igor Mejuev, Akira Kumagai

PFU Limited

Eiichi Kadokura

High Energy Accelerator Research Organization (KEK)

In this work we introduce the results of feasibility study on implementing end-user tailorability in the software for accelerator control system, considering the design and implementation of distributed monitoring application for 12 GeV KEK Proton Synchrotron as an example.

Runtime Tailorability

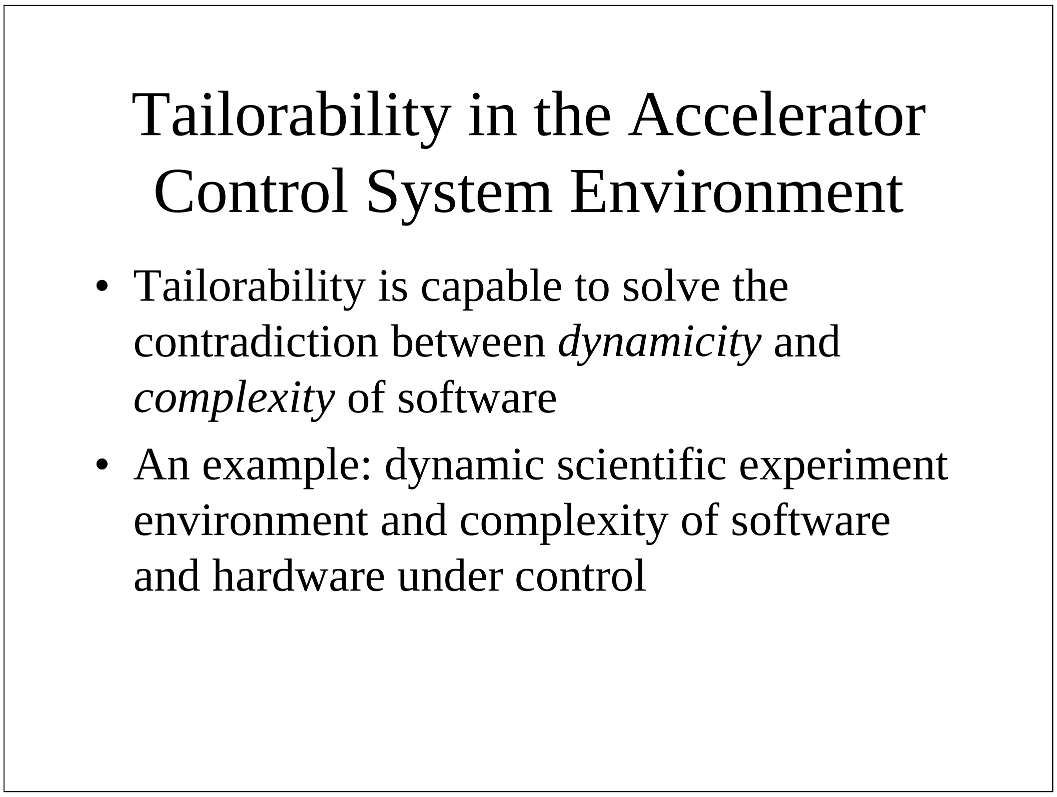
- Tailoring : further evolution of an application *after deployment*
- Motivations : *dynamicity* and *diversity* of requirements
- Dynamicity: waterfall model does not work in the real world
- Diversity: a Web-based system has uncertain and heterogeneous audience

Tailoring is further evolution of an application after deployment in order to adapt it to requirements that were not accounted for in the original design.

Generic motivations for implementing tailorability in software systems are constituted by dynamicity and diversity of users' requirements.

The dynamicity can be derived from the inconsistency between waterfall software development model and the development process taking part in the real world. In accordance with evolutionary and participative design methodology by Floyd et al. the requirements are not given and therefore can not be strictly analyzed.

The diversity of requirements is especially important for the Web-based systems – in general it is difficult to trace the usage of the system deployed on the Web and, moreover satisfy the requirements of heterogeneous groups of users within a single application.



Tailorability in the Accelerator Control System Environment

- Tailorability is capable to solve the contradiction between *dynamicity* and *complexity* of software
- An example: dynamic scientific experiment environment and complexity of software and hardware under control

The application of runtime tailoring can solve the contradiction between dynamicity of requirements and inherent complexity of software present in some application domains.

An accelerator control system is an example of such a domain – the dynamicity and flexibility are the essential requirements for scientific experiment environment, however the amount of hardware and I/O channels involved demands applications of computer control to achieve the consistency of experimental setup.

Contents of This Talk

- Tailoring vs. authoring interface
- VEDICI : a generic implementation framework for tailoring
- Example : remote monitoring application for 12 GeV KEK Proton Synchrotron
- Conclusions and future work

The rest of this talk presents the following:

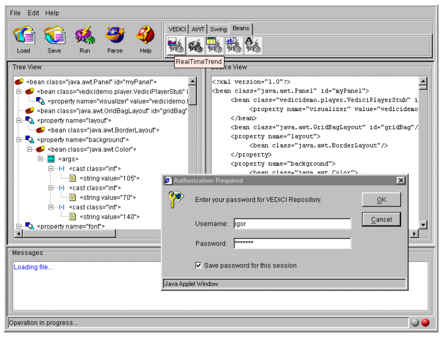
1) Definition of the notion of tailoring interface, considering the differences between tailoring and authoring interfaces

2) Introduction of a generic tailoring platform (VEDICI), which allows integrating multiple tailoring interfaces within a single application instance

3) Design of customizable Web-based application for monitoring beam conditions through the Internet

4) Conclusions and future work

Authoring Interface



```
<bean class="java.awt.Panel" id="myPanel">
  <bean class="vedicidemo.player.VediciPlayerStub" i
    <property name="visualizer" value="vedicidemo.t
  <bean class="java.awt.GridBagLayout" id="gridBag"
  <property name="layout">
    <bean class="java.awt.BorderLayout">
  <property name="background">
    <bean class="java.awt.Color">
      <args>
        <cast class="int">
          <string value="105">
        <cast class="int">
          <string value="70">
        <cast class="int">
          <string value="140">
  <property name="font">
```

```
<?xml version="1.0"?>
<bean class="java.awt.Panel" id="myPanel">
    <bean class="vedicidemo.player.VediciPlayerStub" i
        <property name="visualizer" value="vedicidemo
    </bean>
    <bean class="java.awt.GridBagLayout" id="gridBag"/
    <property name="layout">
        <bean class="java.awt.BorderLayout"/>
    </property>
    <property name="background">
        <bean class="java.awt.Color">
```

Loading file...

Authorization Required

Enter your password for VEDICI Repository.

Username: igor

Password: *******

[✓] Save password for this session

Java Applet Window

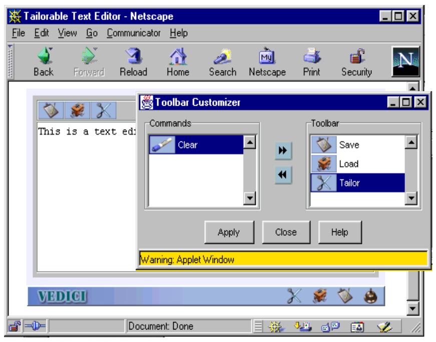
Operation in progress...

- Employed by *developers*
- Requires *full control* of the application
- Assumes distinction between developers and users

An authoring interface is typically used by developers of a computer system. The interface can utilize techniques such as visual manipulations or form-based programming, in order to speed up the process of development. The authoring system is required to provide full control of the application and available APIs, display the composition of the system in a consistent way, and provide integration with runtime and deployment modules.

It is assumed that the users and developers of the system represent distinct and geographically distributed groups.

Tailoring Interface



- Employed by *end-users*
- Applied to the *running application*
- Should reflect users' cognitive views of a given task

The intent of tailoring interface is to provide the users with the possibility to customize an application to their particular needs and work situation. An example of tailoring interface is a text processing application with customizable toolbar.

In the case of tailoring, the modifications should be done by end-users and within the execution environment. For shared applications or applications deployed on the Web the system should support the persistence and authentication of changes made by each user.

The differences between the authoring and tailoring interfaces can be summarized by the following criteria: *audience* (who), *usage pattern* (how) and *context* (when).

VEDICI

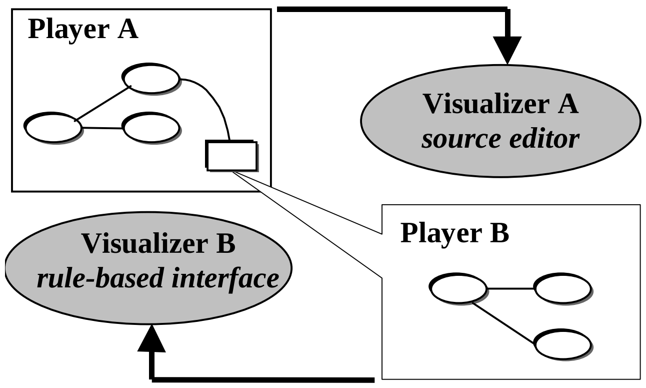
- A generic platform for tailoring of component-based applications
- Allows decoupling of tailoring interfaces and runtime components
- Implemented with Java 1.3 and BML
- Can support multiple tailoring interfaces per application

VEDICI represents a generic tailoring platform which allows decoupling of tailoring interfaces and runtime components.

VEDICI is implemented using Java 1.3 platform and Bean Markup Language (BML).

A VEDICI application is represented as a nested hierarchy of compositional markup specifications with the possibility to associate an individual tailoring component with each specification. This approach allows integrating multiple tailoring interfaces within an application instance.

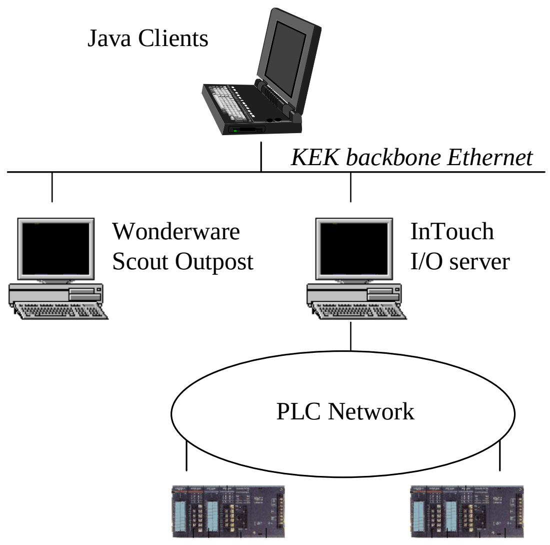
Integrating Multiple Tailoring Interfaces



The basic tailorable unit in our approach is *player* - a wrapper for a composite component with associated *visualizer*. A player compiles a composite component in accordance with a given Compositional Markup Specification. A visualizer defines tailoring interface for a specific class of composite components.

A player can be included into a composite component wrapped by another player, so that a running application represents a nested hierarchy or players. Association of visualizer at the level of individual player allows supporting multiple visualization schemes per application.

Remote Monitoring Application for 12 GeV PS at KEK



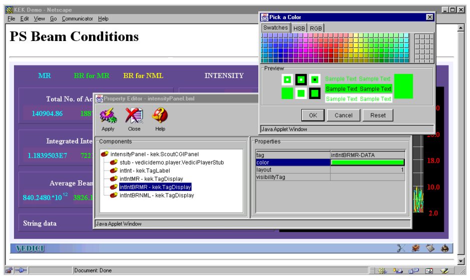
Online monitoring application for KEK Proton Synchrotron should provide display for the beam parameters, accessible in the Java applet environment. The requirements on the design of monitoring application are summarized as follows:

• The applet should be made multiplatform, thus it should not rely on native libraries or OS-specific APIs

• The monitoring system should provide integration with third-party commercial software: Wonderware InTouch, which is widely deployed at KEK Proton Synchrotron

• The required *degree of tailorability* for the monitoring application is identified as the possibility for the end-users to dynamically reassign mappings of GUI components to I/O channels and customize visual preferences (color, layout) for the components. Particular I/O channels are required to display permanently, so that the tailoring functionality should be disabled for the corresponding GUI objects

Implementation



intensityPanel - kek.ScoutCGIPanel
- stub - vedicidemo.player.VediciPlayerStub
- intInt - kek.TagLabel
- intIntMR - kek.TagDisplay
- intIntBRMR - kek.TagDisplay
- intIntBRNML - kek.TagDisplay

tag	intIntBRMR-DATA
color	🟩
layout	1
visibilityTag	

- Compositional: Java Beans + integrating BML scripts
- Integrated with Wonderware InTouch
- Applications can be tailored using a mixture of reusable visualizers
- Application repository with authorization by username and password

The monitoring application is composed from Java Beans integrated by nested XML-based scripts (BML).

The data update is performed by a dedicated component, which wraps Scout Outpost CGI interface and provides a refresh manager for dynamic monitoring components. The refresh manager performs data polling by sending batch requests to the Scout Outpost server.

The application provides an authoring tool for developing new applications and reusable visualizers applicable for customization of the applications at runtime (*HierarchyBrowser*, *PropertyEditor*, *SourceEditor*, *VisualizerStub*).

The users can save the customized applications in a server-side repository with authorization by username and password. After restart of browser the changes can be recovered using the same authentication scheme.

- Conclusions
  - Implementation of runtime tailorability in the accelerator control domain is feasible
  - Preferred implementation technique is *tailorability by integration* (component-based)
- Future Work
  - Rich set of components
  - Server push communication model
  - Tailorable control applications

The prototype implementation showed that tailorability by integration can be implemented in the accelerator control environment using Java Beans and compositional markup specifications (BML).

In the future we would like to extend the prototype implementation by providing a rich set of components and custom visualizers, usable in the accelerator control system domain.

The current implementation uses polling of CGI server that creates redundant traffic in the laboratory network. In the future version we consider replacing the polling with server push interface, which is based on the existing portable implementation of shared data channels.

Trademarks