JCE
A Java-based Commissioning Environment tool

Hiroyuki Sako, JAEA
(hiroyuki.sako@j-parc.jp)
Hiroshi Ikeda, Visible Information Center Inc.
SAD Workshop 06, 6 Sep 2006

- Motivation and background
- Status of development
- Summary
- Demo

Advantages of SAD as a beam commissioning tool

- Seamless handlings of device control, data monitoring, model calculations, in a unified platform
- Flexible interpreter language SAD script
  - Convenient as commissioning language since it runs without compilation (quick developing a tool and easy debugging). Successful in KEKB

⇒In J-PARC we considered SAD as a primary candidate of commissioning tool

To use SAD;

- It is necessary to maintain (and debug) the SAD codes
- Addition of J-PARC own functionalities is necessary
  - Controls, modeling, data analysis

  - For these, we needed to understand the details of the codes and also make necessary modifications for easy extension and maintenance scheme

Looking into SAD code

We struggled to read and understand the codes for a few months.

But, it is found to be very hard;

- Highly technical Fortran codes to analyze SAD script syntax
- Use of common blocks and indexes to arrays
- Documents and comments are not enough

- Script analysis parts and core parts are interlaced deeply and hard to separate

↓

A completely new tool easy to understand in a modern OO technology is a solution

(JAVA Commissioning Environment = JCE)

Java Commissioning Environment (JCE)

Unified commissioning environment tool with Java

Primary Goals: To support users to write commissioning tools in SAD script
with simple scheme to maintain codes
with easy addition of commands/functionalities

Inheriting SAD concepts and SAD script language

- Effective commissioning by seamless handlings of device control, data monitoring, model calculations, GUI, and analysis tools

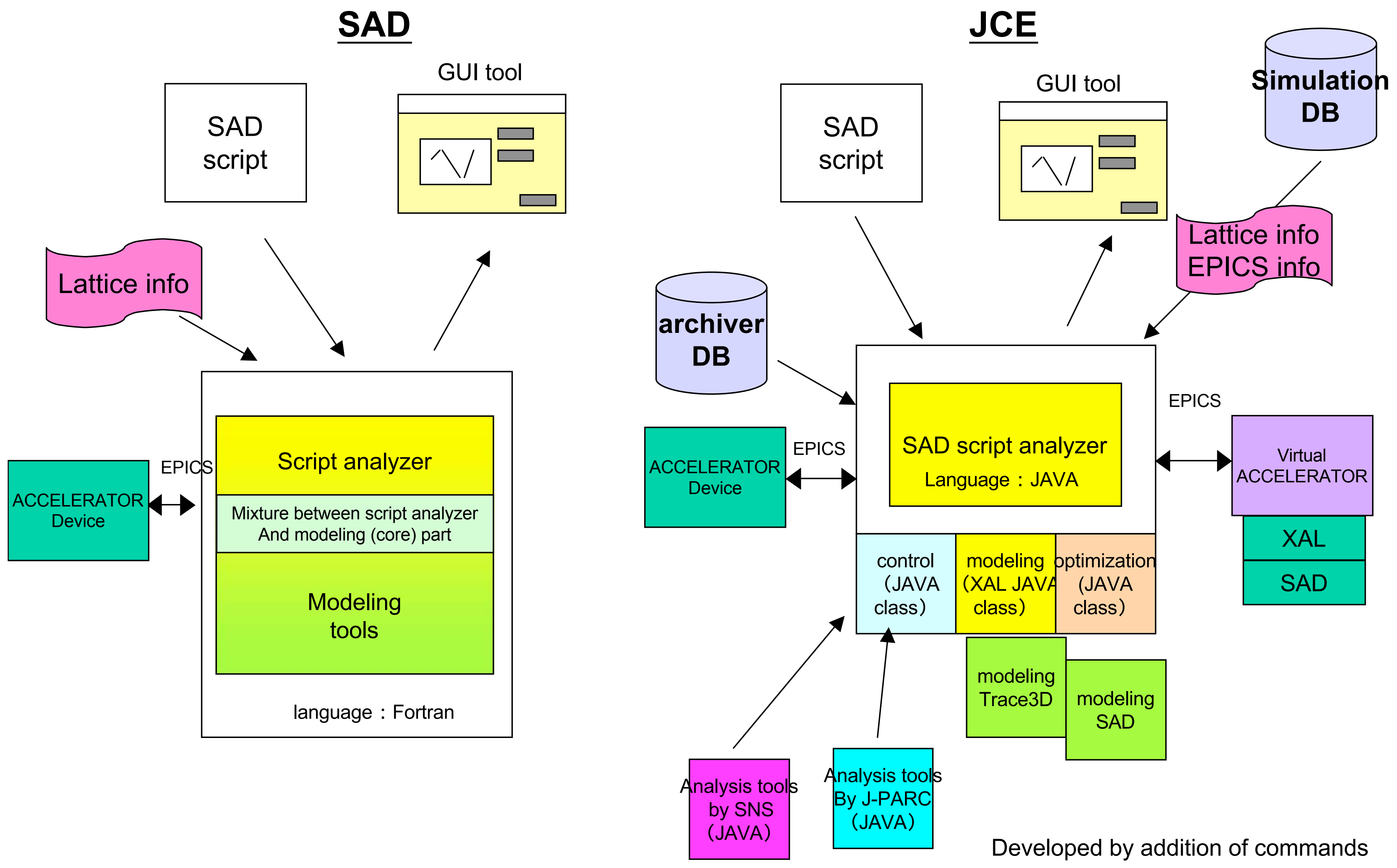
– New features and advantages of JCE
  - Java Implementation
    - Multi-platform
      - LINUX, Windows,… wherever with JAVA
    - Easy to install and run
      - Just unzip and run
  - We can maintain and develop codes by ourselves for J-PARC
  - Manuals and documents of code explanations
  - Utilization of Java libraries in the world

– Interface to RDB, and data query functions will be implemented
  - Device parameter DB, data archiver DB, operation parameter DB (Save & Restore DB)
  - Via JDBC (Java DB library)

– Inclusion of control tools being developed in control group

– Modeling (XAL, Trace3D, (SAD, MAD))
  - Call of SAD and MAD as external simulator possible

Architecture Comparison



Status of JCE Development

- About 300 functions implemented so far
- Functions same as SAD
  - Flow control
    - If, For, Do, While,…
  - List operations
    - Table, Flatten, Thread,…
  - File I/O
    - Open, Read, Write
  - GUI components, graph
    - Frame, Window, Button, TextLabel,…
  - EPICS I/O
    - CaRead, CaWrite, CaMonitor
- Not supported
  - Non-Mathematica syntax (SAD model core)
- New features (planned)
  - RDB I/O
  - plug-in of user defined Java classes
- Model calculations with Mathematica like functions
  - XAL (directly) and Trace3D (via JNI (Java Native Interface))
  - SAD and MAD as external commands
  - Automatic generation of input files for these models from a common Simulation DB is established.

XAL library in JCE

A high level application framework developed in SNS

- Successful in SNS commissioning of LINAC and ring
- Java implementation in well organized structure
  - Easy to include in JCE
- Commissioning, modeling, EPICS, RDB tools

– XAL Online Modeling

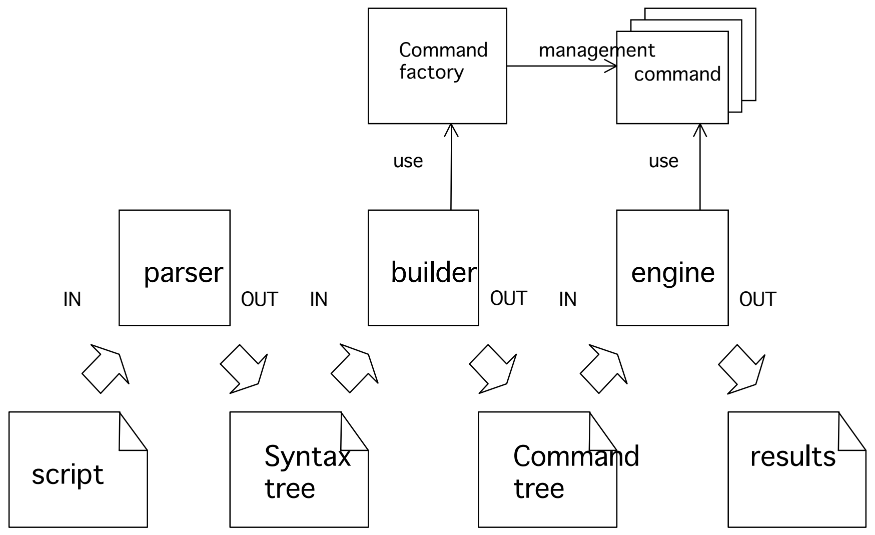
- Space-charge envelope simulation with ellipsoid space-charge distribution
- Functions for J-PARC LINAC (features in Trace3D)
- Ring modeling functions are being developed for SNS commissioning (envelope calculations, closed orbits, single particle tracking)

JCE algorithm flow

1. Parser: creates a syntax tree

   By JavaCC (Java Compiler-Compiler)

2. Builder: convert the syntax tree to a command tree
3. Evaluation engine: execute the results using command implementation



JCE code organization

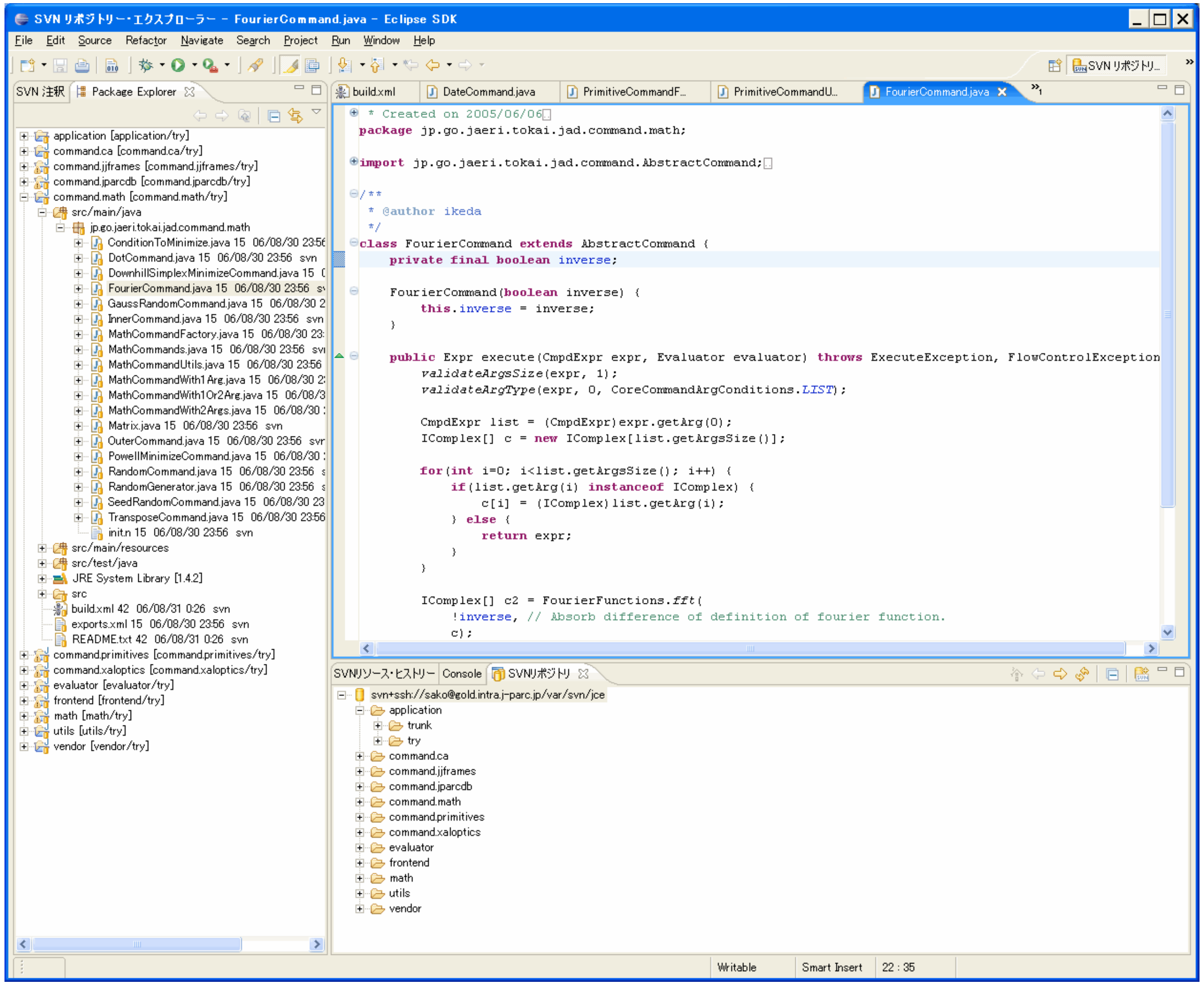
The JCE codes are organized in the following packages

Clear separation of interpreter analysis part (evaluator) and command implementation (command.*) achieved!

• vendor	External Java libraries
• utils	General utilities
• math	Mathematic utilities
• evaluator	Evaluation engine (script analysis part)
• command.*	Implementation of each command class (different packages for each category)
• frontend	UI for execution of application
• application	Construction of application

JCE development

- Use of eclipse for IDE
- Code version management with SVN (sub version)



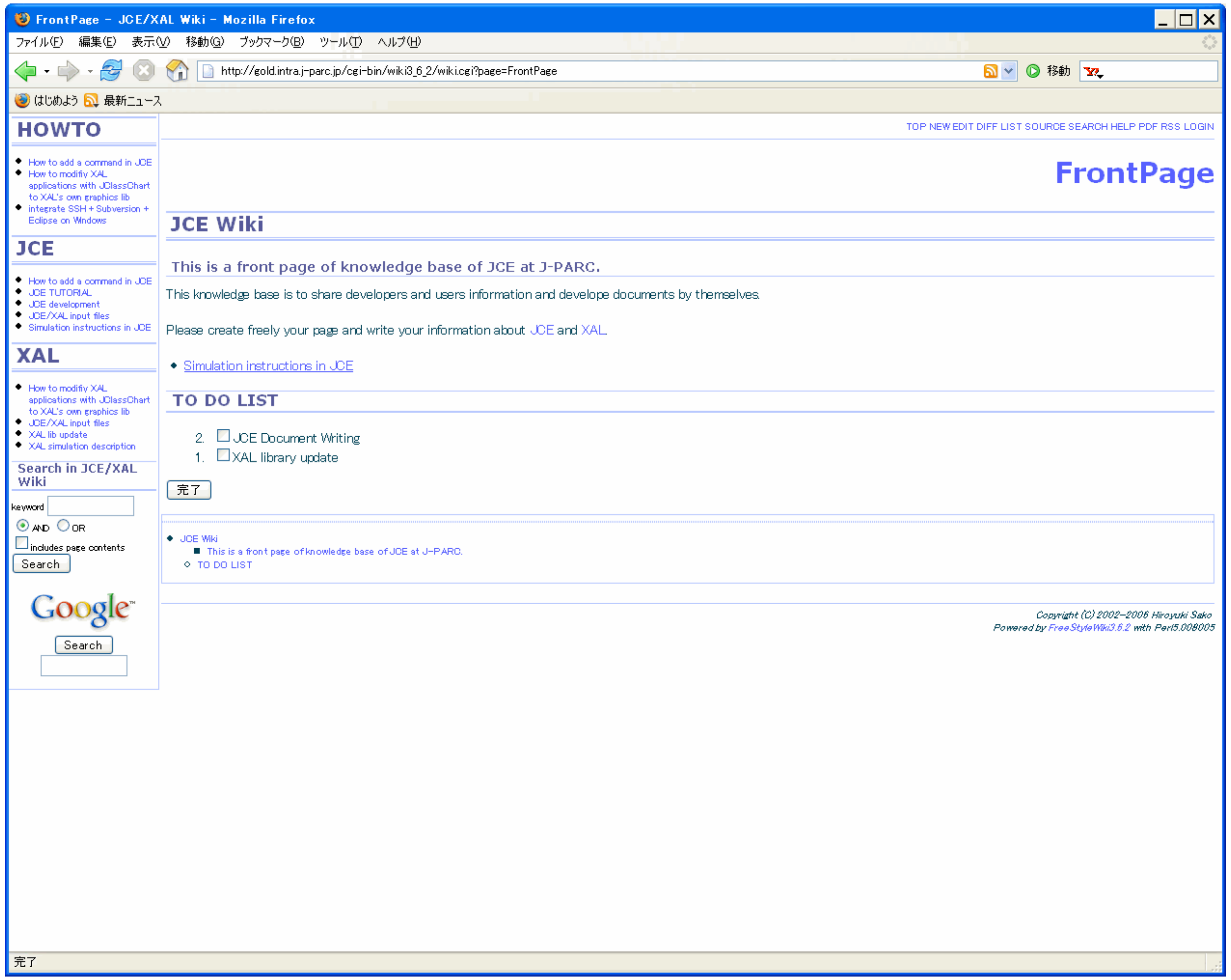
```
application [application/try]
command.ca [command.ca/try]
command.jjframes [command.jjframes/try]
command.jparcdb [command.jparcdb/try]
command.math [command.math/try]
  src/main/java
    jp.go.jaeri.tokai.jad.command.math
      ConditionToMinimize.java 15  06/08/30 23:56
      DotCommand.java 15  06/08/30 23:56  svn
      DownhillSimplexMinimizeCommand.java 15
      FourierCommand.java 15  06/08/30 23:56  s
      GaussRandomCommand.java 15  06/08/30 2
      InnerCommand.java 15  06/08/30 23:56  svn
      MathCommandFactory.java 15  06/08/30 23:
      MathCommands.java 15  06/08/30 23:56  sv
      MathCommandUtils.java 15  06/08/30 23:56
      MathCommandWith1Arg.java 15  06/08/30 2
      MathCommandWith1Or2Arg.java 15  06/08/3
      MathCommandWith2Args.java 15  06/08/30
      Matrix.java 15  06/08/30 23:56  svn
      OuterCommand.java 15  06/08/30 23:56  svr
      PowellMinimizeCommand.java 15  06/08/30
      RandomCommand.java 15  06/08/30 23:56  s
      RandomGenerator.java 15  06/08/30 23:56  s
      SeedRandomCommand.java 15  06/08/30 23
      TransposeCommand.java 15  06/08/30 23:56
      init.n 15  06/08/30 23:56  svn
  src/main/resources
  src/test/java
  JRE System Library [1.4.2]
  src
  build.xml 42  06/08/31 0:26  svn
  exports.xml 15  06/08/30 23:56  svn
  README.txt 42  06/08/31 0:26  svn
command.primitives [command.primitives/try]
command.xaloptics [command.xaloptics/try]
evaluator [evaluator/try]
frontend [frontend/try]
math [math/try]
utils [utils/try]
vendor [vendor/try]
```

```java
 * Created on 2005/06/06
package jp.go.jaeri.tokai.jad.command.math;

import jp.go.jaeri.tokai.jad.command.AbstractCommand;

/**
 * @author ikeda
 */
class FourierCommand extends AbstractCommand {
    private final boolean inverse;

    FourierCommand(boolean inverse) {
        this.inverse = inverse;
    }

    public Expr execute(CmpdExpr expr, Evaluator evaluator) throws ExecuteException, FlowControlException
        validateArgsSize(expr, 1);
        validateArgType(expr, 0, CoreCommandArgConditions.LIST);

        CmpdExpr list = (CmpdExpr)expr.getArg(0);
        IComplex[] c = new IComplex[list.getArgsSize()];

        for(int i=0; i<list.getArgsSize(); i++) {
            if(list.getArg(i) instanceof IComplex) {
                c[i] = (IComplex)list.getArg(i);
            } else {
                return expr;
            }
        }

        IComplex[] c2 = FourierFunctions.fft(
            !inverse, // Absorb difference of definition of fourier function.
            c);
```

```
svn+ssh://sako@gold.intra.j-parc.jp/var/svn/jce
  application
    trunk
    try
  command.ca
  command.jjframes
  command.jparcdb
  command.math
  command.primitives
  command.xaloptics
  evaluator
  frontend
  math
  utils
  vendor
```

JCE Wiki Page (under development)

- Manuals
- How-to
- Tutorials
- Development logs



FrontPage - JCE/XAL Wiki - Mozilla Firefox

ファイル(F) 編集(E) 表示(V) 移動(G) ブックマーク(B) ツール(T) ヘルプ(H)

http://gold.intra.j-parc.jp/cgi-bin/wiki3_6_2/wiki.cgi?page=FrontPage

はじめよう 最新ニュース

HOWTO

- How to add a command in JCE
- How to modifiy XAL applications with JClassChart to XAL's own graphics lib
- integrate SSH + Subversion + Eclipse on Windows

JCE

- How to add a command in JCE
- JCE TUTORIAL
- JCE development
- JCE/XAL input files
- Simulation instructions in JCE

XAL

- How to modifiy XAL applications with JClassChart to XAL's own graphics lib
- JCE/XAL input files
- XAL lib update
- XAL simulation description

Search in JCE/XAL Wiki

keyword [✓ AND] OR

includes page contents

Search

Google

Search

TOP NEW EDIT DIFF LIST SOURCE SEARCH HELP PDF RSS LOGIN

FrontPage

JCE Wiki

This is a front page of knowledge base of JCE at J-PARC.

This knowledge base is to share developers and users information and develope documents by themselves.

Please create freely your page and write your information about JCE and XAL.

- Simulation instructions in JCE

TO DO LIST

2. ☐ JCE Document Writing
1. ☐ XAL library update

完了

- JCE Wiki
  - This is a front page of knowledge base of JCE at J-PARC.
  - TO DO LIST

Copyright (C) 2002-2006 Hiroyuki Sako
Powered by FreeStyleWiki3.6.2 with Perl5.008005

完了

Summary and perspectives

- JCE has been developed in Java for simple maintenance scheme and extensibility but keeping the concepts of SAD as a powerful commissioning platform
- We start developing tools for LINAC commissioning in Dec

Demo script of a simple orbit correction in LINAC-MEBT1

- MEBT1: a beam transport line with 10 steering dipole magnets, 8 quadrupoles and 8 BPM’s
- Initial beam position has offset
  - Orbit has deviations
- Calculate optimum steering magnet settings to correct orbit deviations
  - With Simplex minimization algorithm
- Set the obtained magnet parameters to a virtual accelerator via EPICS and monitor the BPM position data