Aircrew
Brian Binnie Pete Siebold
Mike Melvill Doug Shane

Scaled Composites, LLC
1624 Flight Line, Mojave CA 93501
Info@scaled.com 661-824-4541

Development of a state machine sequencer for the Keck Interferometer: evolution, development & lessons learned using a CASE tool approach

Leonard J. Reder, Andrew Booth,
Jonathan Hsieh, Kellee Summers

June 21, 2004

Leonard.J.Reder@jpl.nasa.gov

Jet Propulsion Laboratory
California Institute of Technology

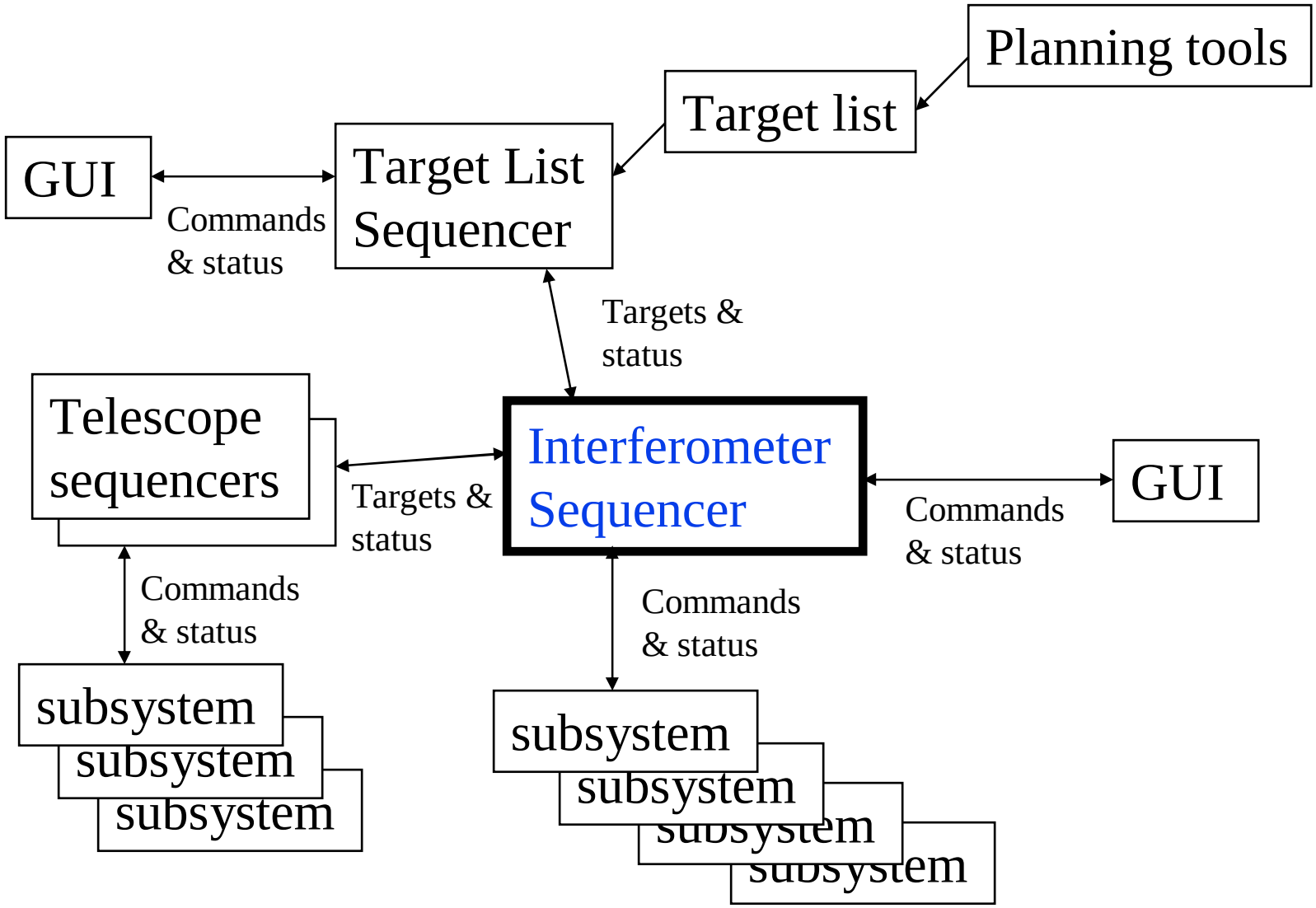
Keck Interferometer

- 2 Keck 10m telescopes with full adaptive optics
- Wavelengths 1.2 um to 10 um
  (AO operates in visible)
- Science: nulling, differential phase, astrometry, imaging
- First fringes with Keck Telescopes March, 2001

What I will talk about

- Interferometer (IF) Sequencer within overall software
- What is the CASE Tool?
- Interferometer Major Components
- Rhapsody and IF Sequencer Design
  - Static Class and Harel Statechart Diagrams
- Lessons Learned

Target/Observation Sequencing Hierarchy

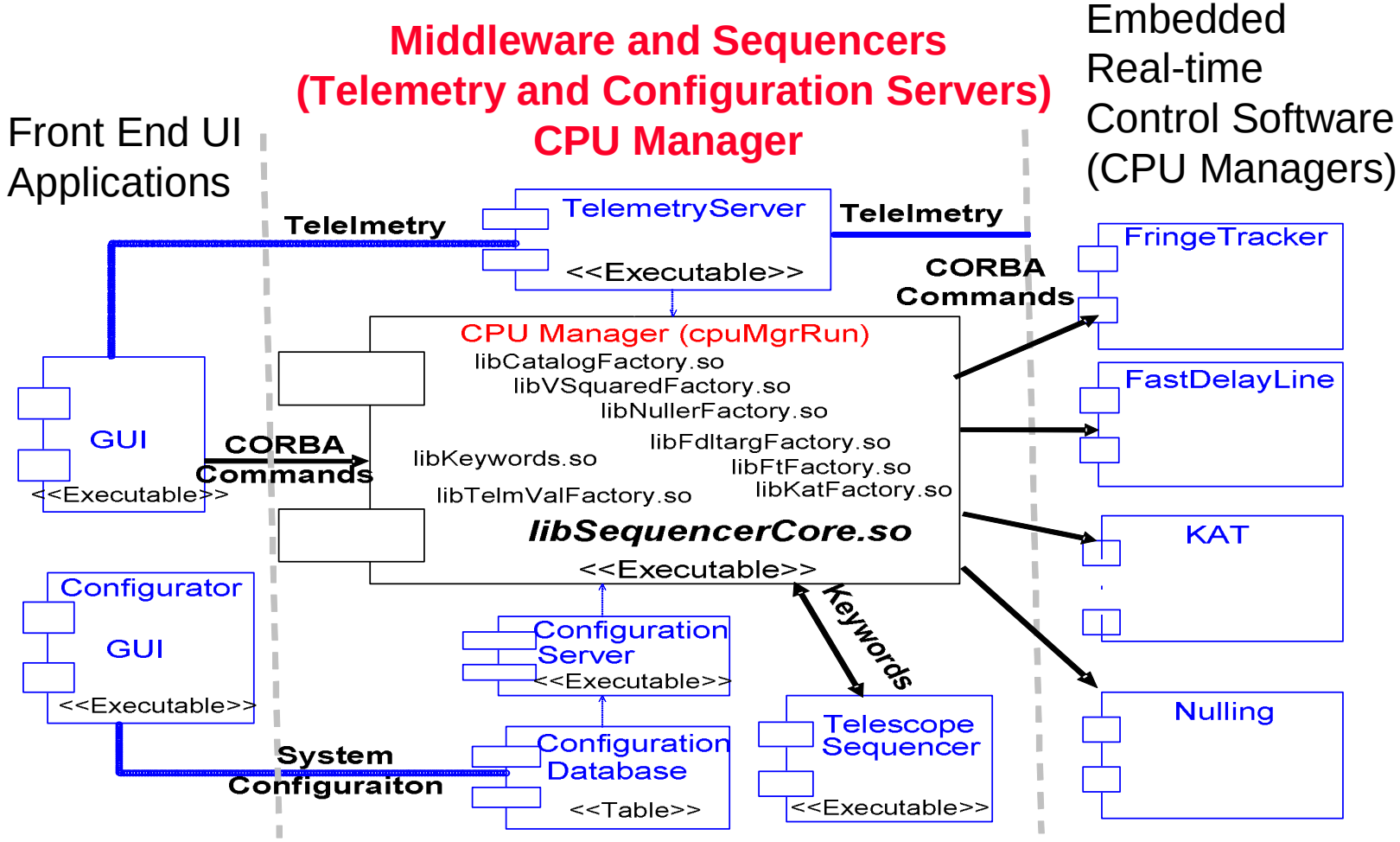


KECK INTERFEROMETER PROJECT

Interferometer Sequencer

- Provide high-level control for observing
  - Send commands via CORBA to JPL RTC software
  - Control Keck Telescope Sequencers via Keywords API
  - Support Science modes: nulling, differential phase, astrometry, imaging (multi-baseline)
- ***State-machine classes running in separate threads***
- Multi-mode Science Ops
  - CPU Manager allows load, instantiate and desroy objects
  - Utilized Rhapsody internal shared infrastructure for state machine to state machine event passing
  - Configurable from common RTC database
  - New more flexible Python GUI using CORBA

Component diagram of IF Sequencer Configuration with JPL RTC Toolkit



KECK INTERFEROMETER PROJECT

KECK INTERFEROMETER PROJECT

Rhapsody: What is the tool?

- It is a CASE (Computer Aided Software Engineering) Tool

- One uses UML (Unified Modeling Language) to design object structures and state machines

- The tool generates C++ code from the UML

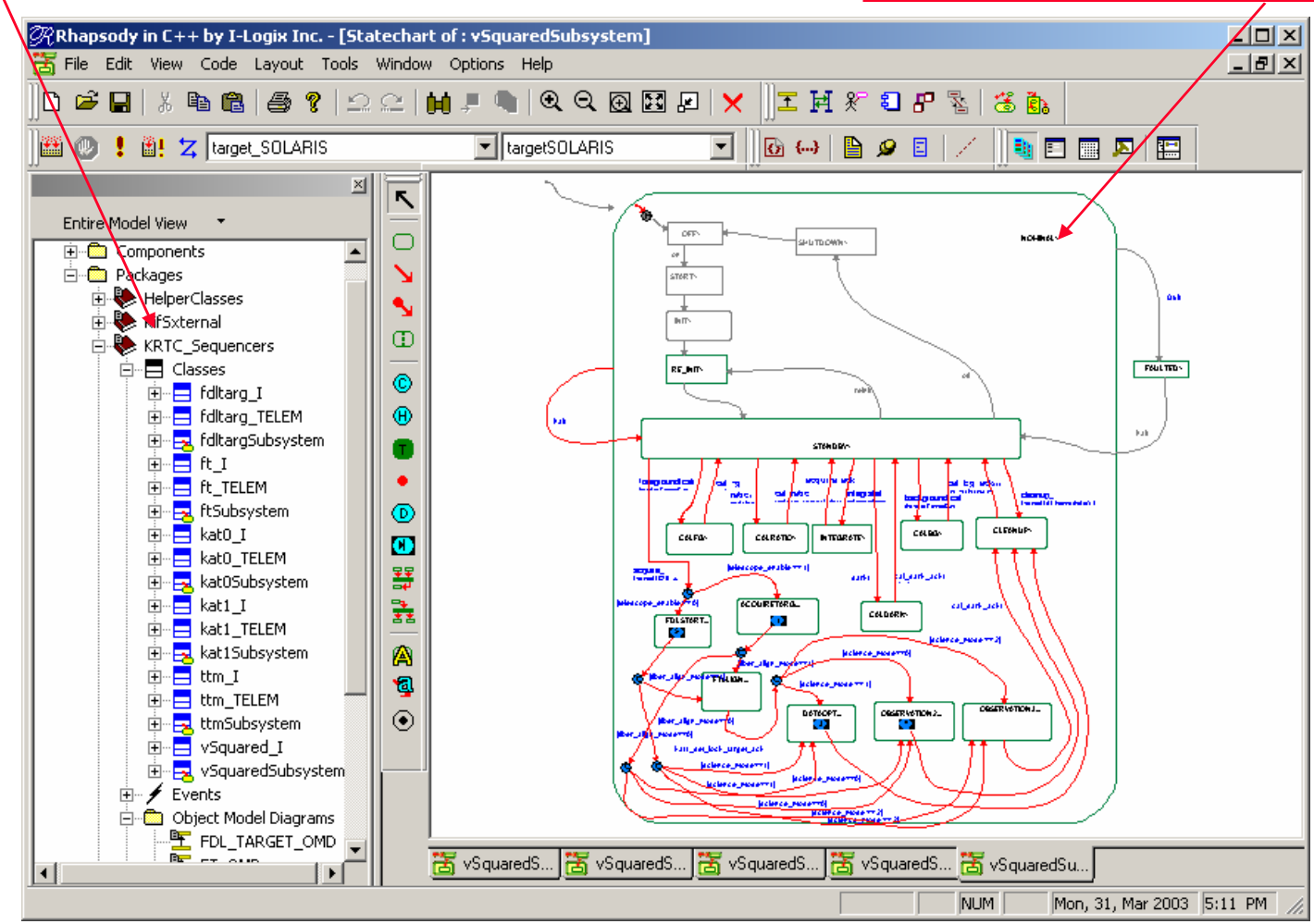
- **Our main use of Rhapsody is to automatically generate multi-threaded state machine code!**

- One can specify state machine behavior for some classes using UML style state machine diagrams

Screenshot showing Rhapsody CASE Tool

Model View

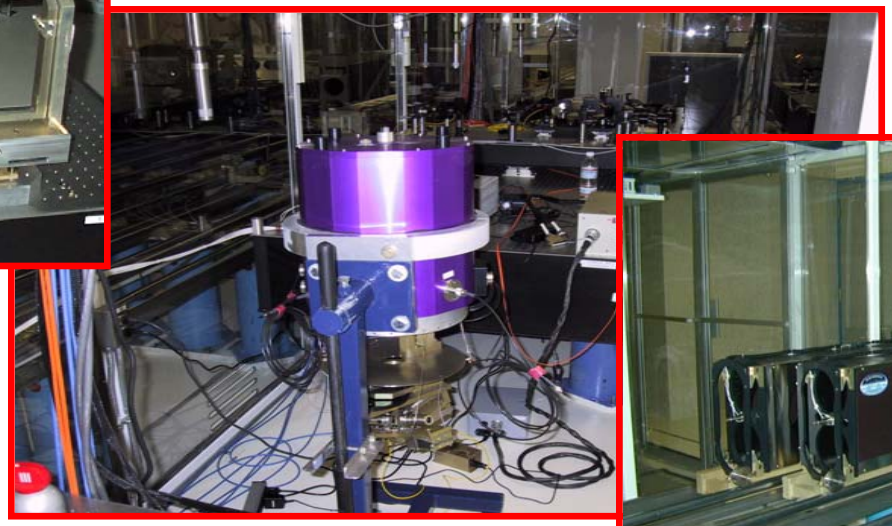
State Machine or Object Model Graphical Views



KECK INTERFEROMETER PROJECT

Interferometer Major Components

Beam Transport Optics (Coude Train)



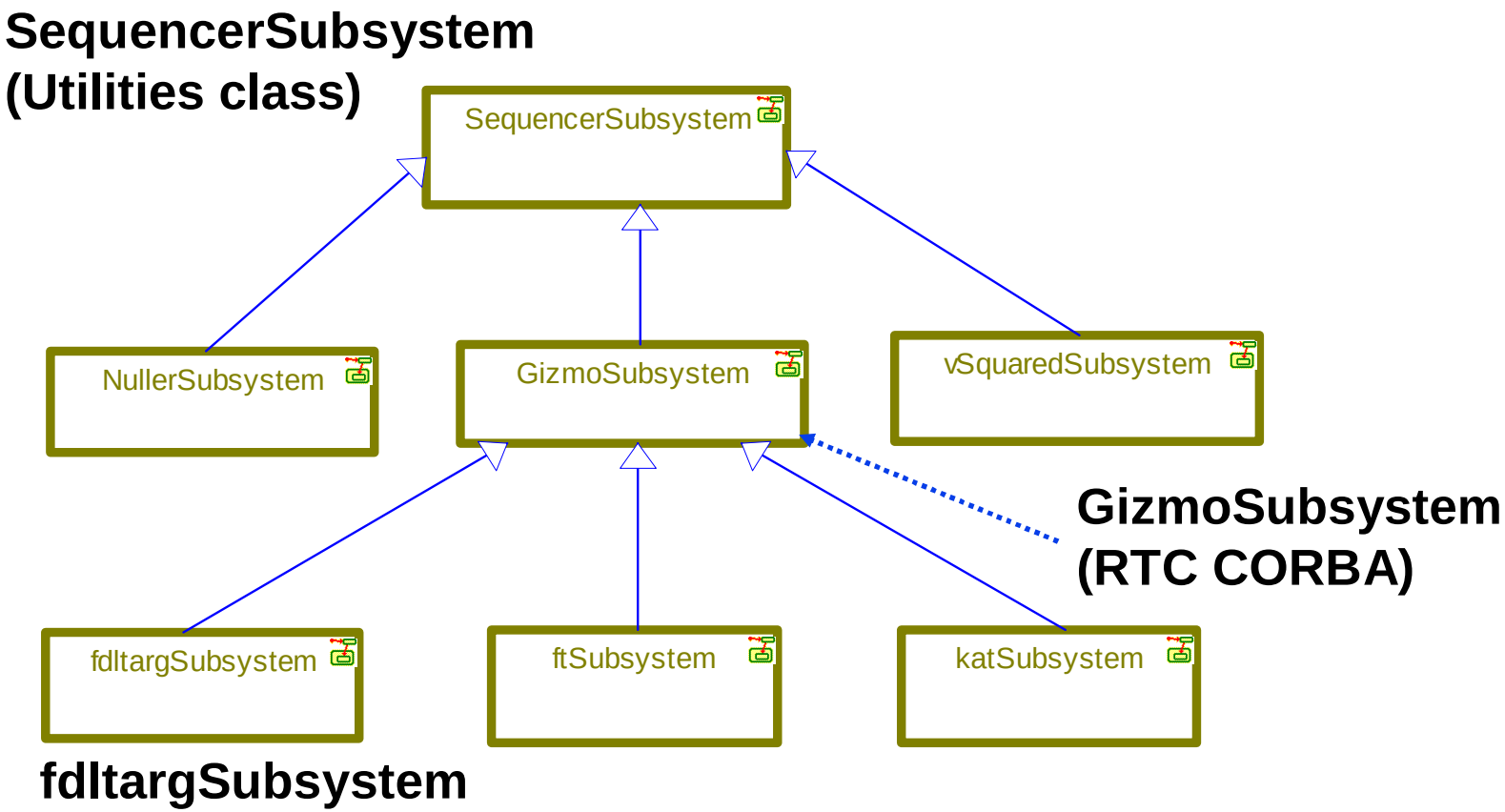
Fringe Tracker (FATCAT)

Fast Delay Lines (FDL)

Angle Tracker (KAT)

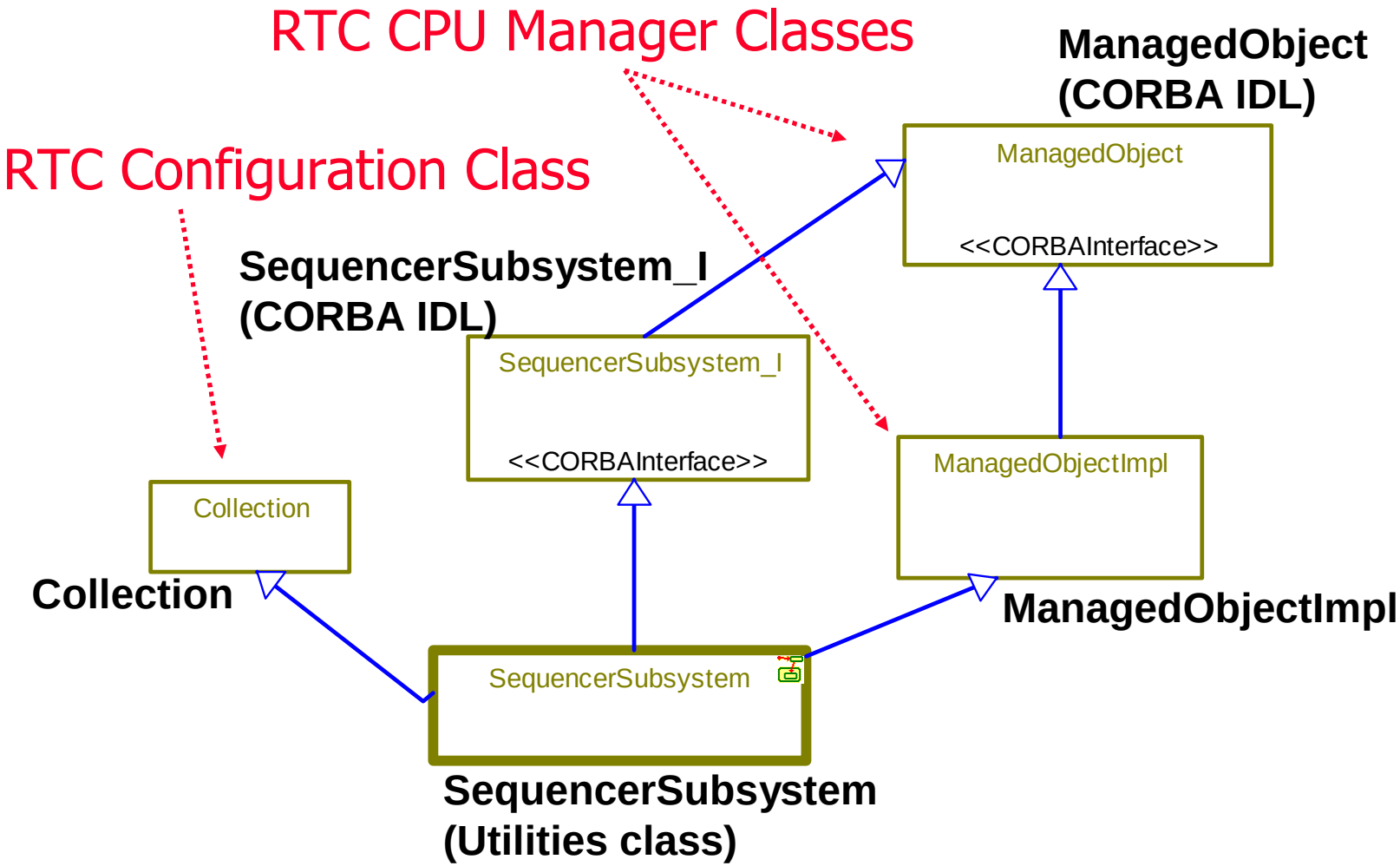
Long Delay Lines (LDL)

Static Class Diagram of Subsystems Class Hierarchy



KECK INTERFEROMETER PROJECT

SequencerSubsystem Utility Class Relationship to RTC Toolkit

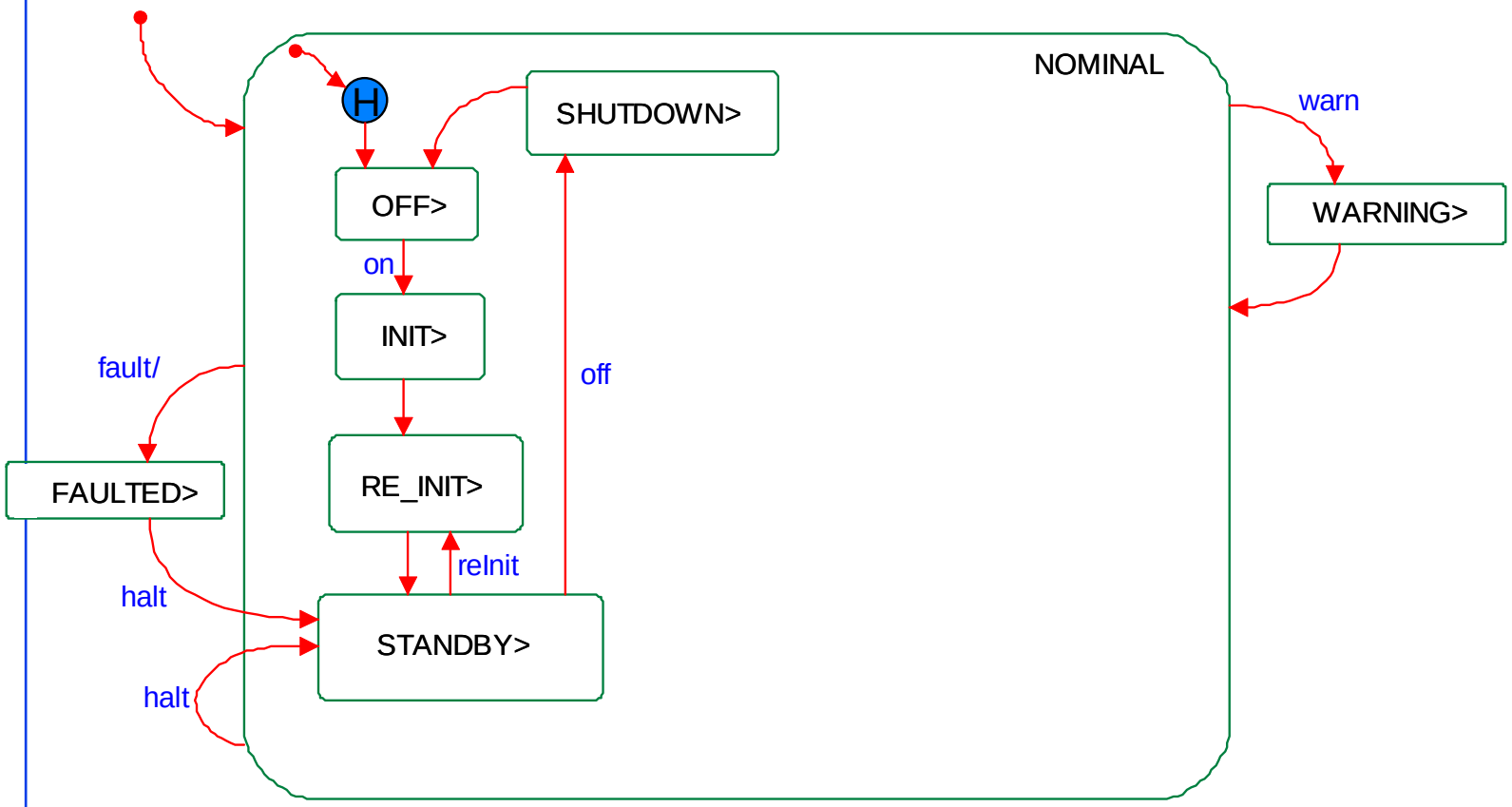


KECK INTERFEROMETER PROJECT

Harel State Machine-Based Design (Class)

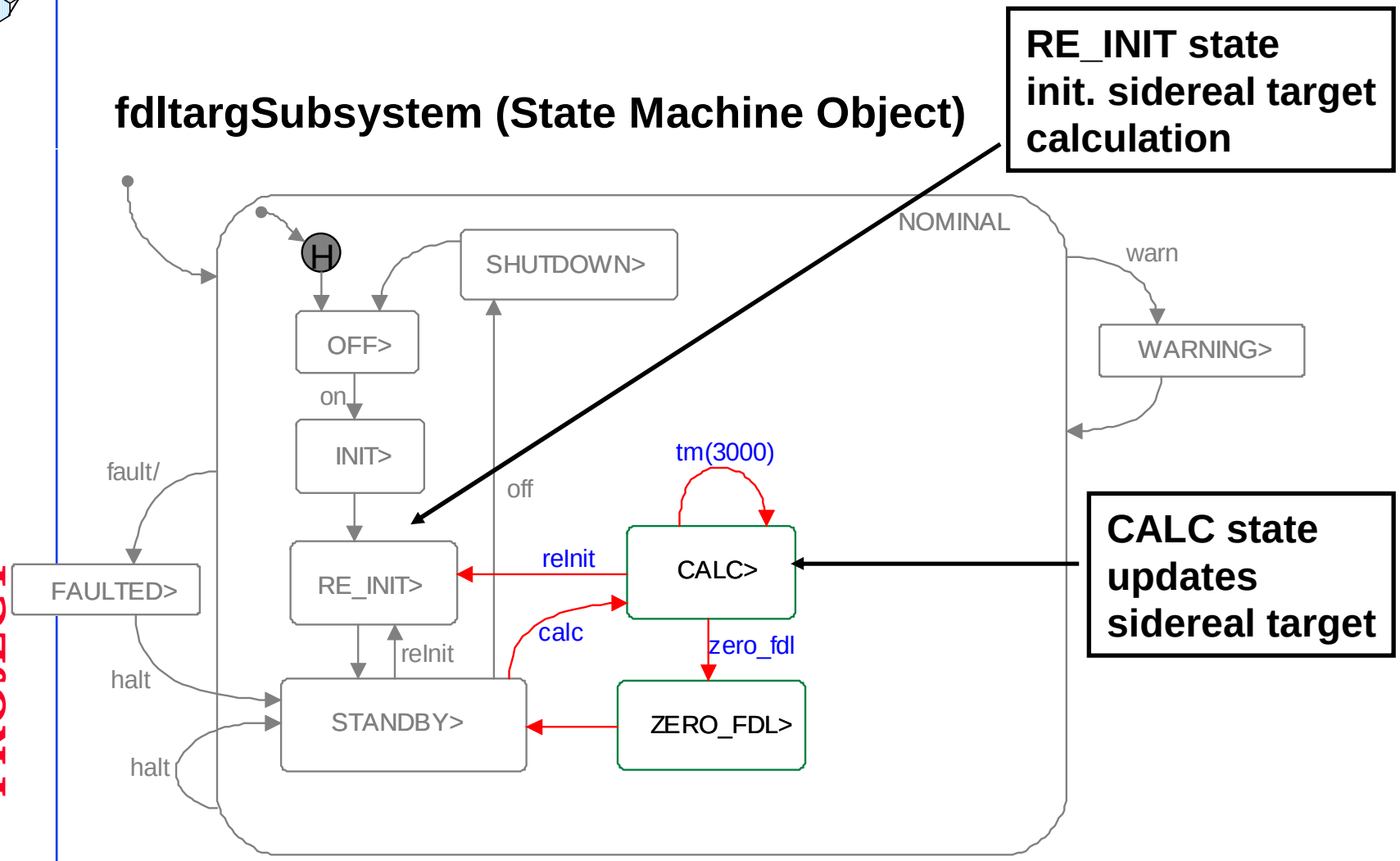
- All state machines inherit this class

SequencerSubsystem (Utilities class)



Fast Delay Line Target Generator
Harel State Machine

fdltargSubsystem (State Machine Object)

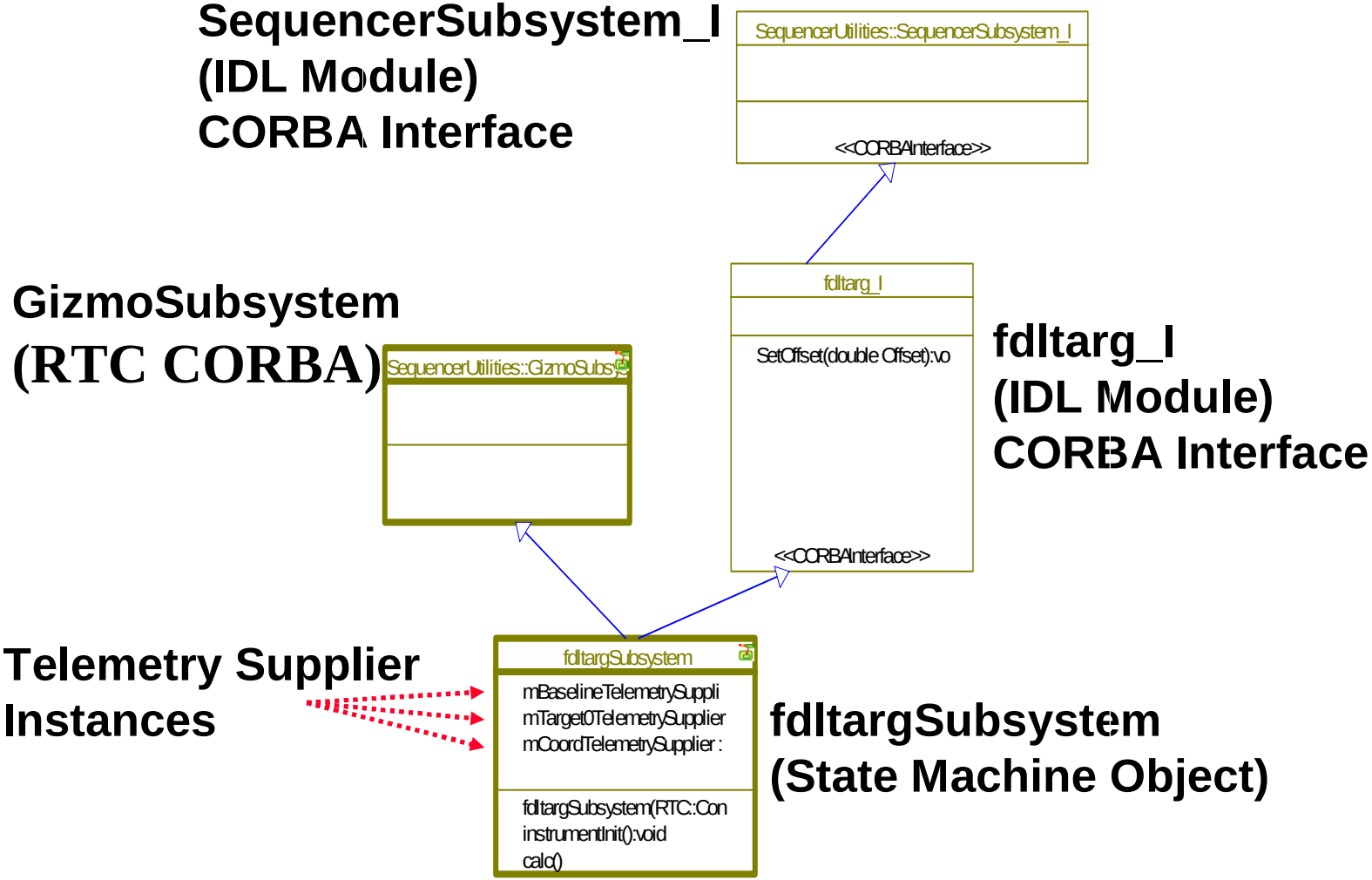


RE_INIT state init. sidereal target calculation

CALC state updates sidereal target

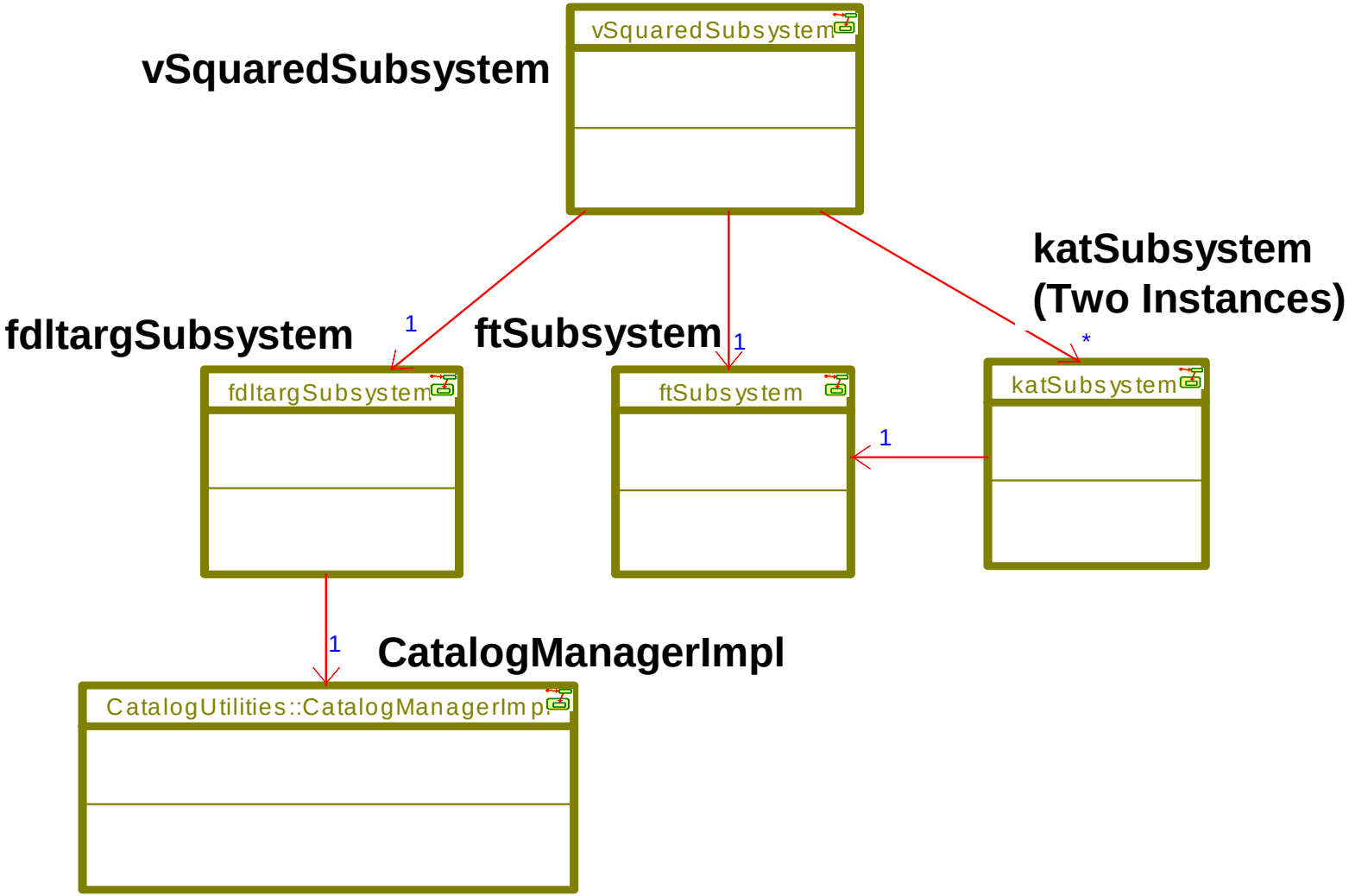
KECK INTERFEROMETER PROJECT

Fast Delay Line Target Generator (fdltargSubsystem) Static Class Diagram

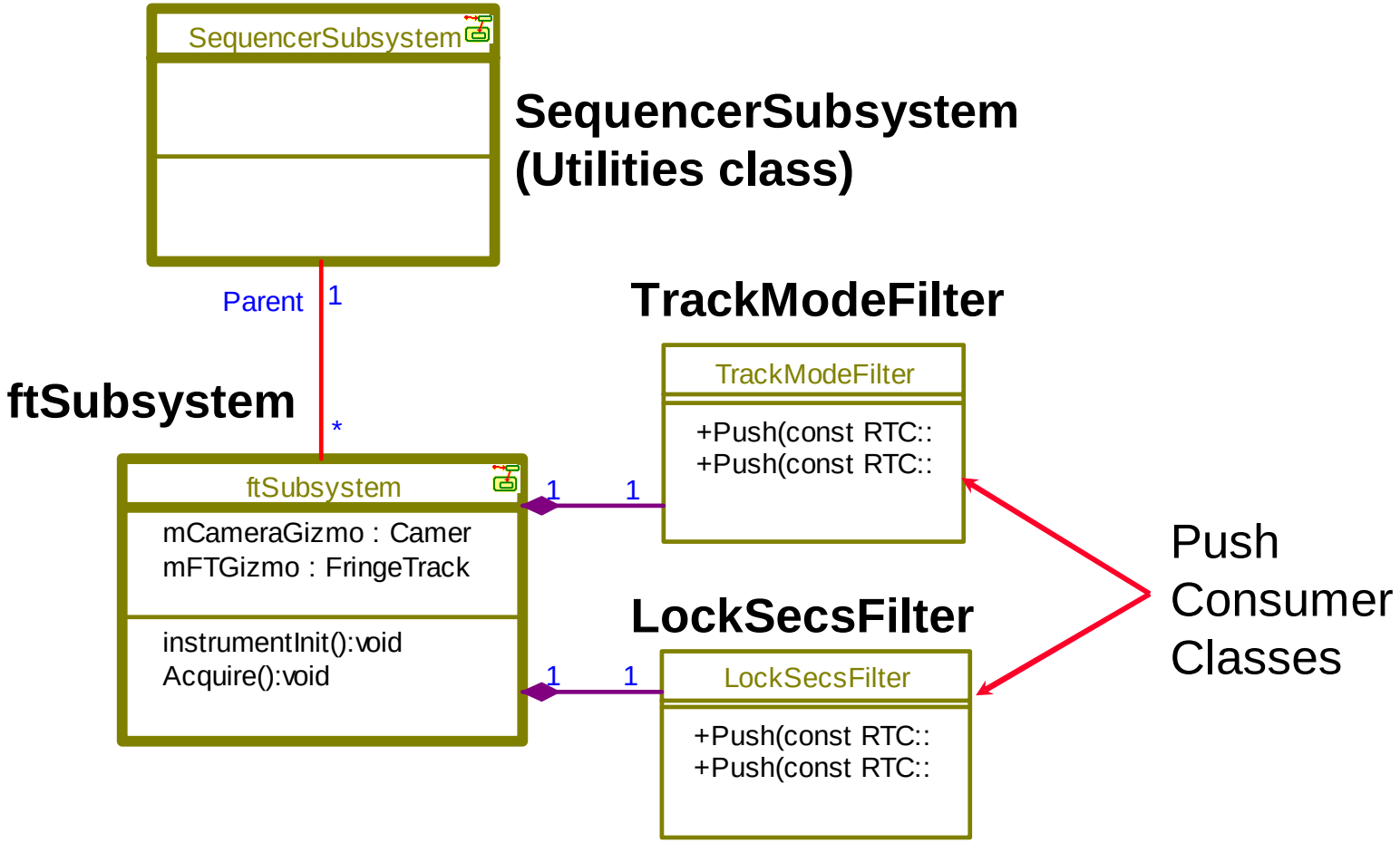


KECK INTERFEROMETER PROJECT

Static Class Diagram of Subsystem Associations for Visibility Operation



Fringe Tracker UML
(Shows Telemetry Consumers)



KECK INTERFEROMETER PROJECT

Lessons Learned (1)

1) For CASE tool to be significantly beneficial requires an overall commitment and investment.

a. Consider a process for development and then a tool.

2) Hard to integrate multiple infrastructures!

a. Even harder to integrate them into a CASE tool.
b. Ideally, on new projects try not to mix infrastructures.
c. Legacy code is inevitable so minimize mixing.

3) "Old programming habits die hard" [Douglas Schmidt, UCLA Ext. 2002].

a. It is hard to learn graphical/coding UML methodology of the CASE tool environment if you have been a line-by-line coder for years.

Lessons Learned(2)

4) Maintainability of code is vastly improved.

a. C++ code is coupled to a UML graphical model so it is more understandable.

b. Automatically limits the amount of reverse engineering of code required later in software lifecycle!

5) Standardized auto-generated code leads to better infrastructure.

a. Design can be rapidly changed graphically with only a minimum amount of coding required.

6) Templates were not included with the Rhapsody case tool so we implemented them externally and hooked them into the tool. This is a kludge and is confusing to a new developer. This effectively defeats rule 4 above.

Lessons Learned (3)

7) The urge to use many of Rhapsody's framework features has sometimes overwhelmed the wiser inclination to use more standardized tools such as the STL (C++ Standard Template Library).

8) Features in a tool may look good during a sales demo, but be careful!

9) When starting, one should evaluate and not guess at a CASE tool to use; there are lots of choices today.

10) Version control of UML models can be problematic even between only a few developers.

Summary

- Implemented a multi-threaded state machine based, high-level, control capability for the Keck Interferometer

- State-machine classes have been designed to be highly configurable and flexible while reusing a basic state-machine initialization and error handling scheme

- CASE tool approach is superior to hand line-by-line coding
  - Improved maintenance over the life-cycle is realized
  - Dealing with third party infrastructures outside of the Rhapsody tool is problematic at first
  - After significant effort at integration it becomes easy to graphically add states and additional software architecture

- Rhapsody is an excellent tool, especially for state machine design and the problem fits nicely into the infrastructures provided by I-logix and the JPL developed RTC toolkit

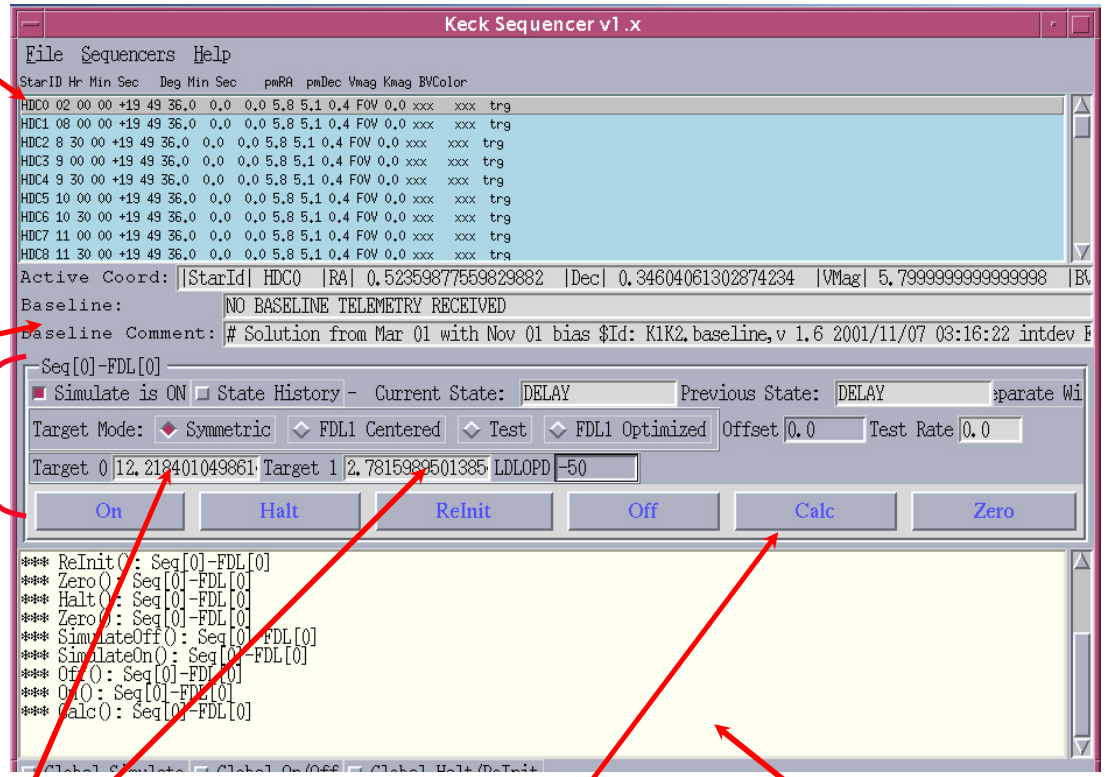
Interferometer Sequencer User Interface

KECK INTERFEROMETER PROJECT

Selected observation record

Current observation & baseline vector

Optional State Machine Control Panel



Generated Delay Line Sidereal Targets

Sequencer observing controls

Text Status