XAL Online Model Enhancements for J-PARC Commissioning and Operation*

CHRISTOPHER K. ALLEN, ORNL
HIROYUKI SAKO, JAEA
MASANORI IKEGAMI, KEK
GUOBAO SHEN, JAEA
HIROSHI IKEDA, VIC
TOMOHIRO OHKAWA, JAEA
AKIRA UENO, JAEA

*Work supported by KEK under a foreign visiting researcher grant

Abstract

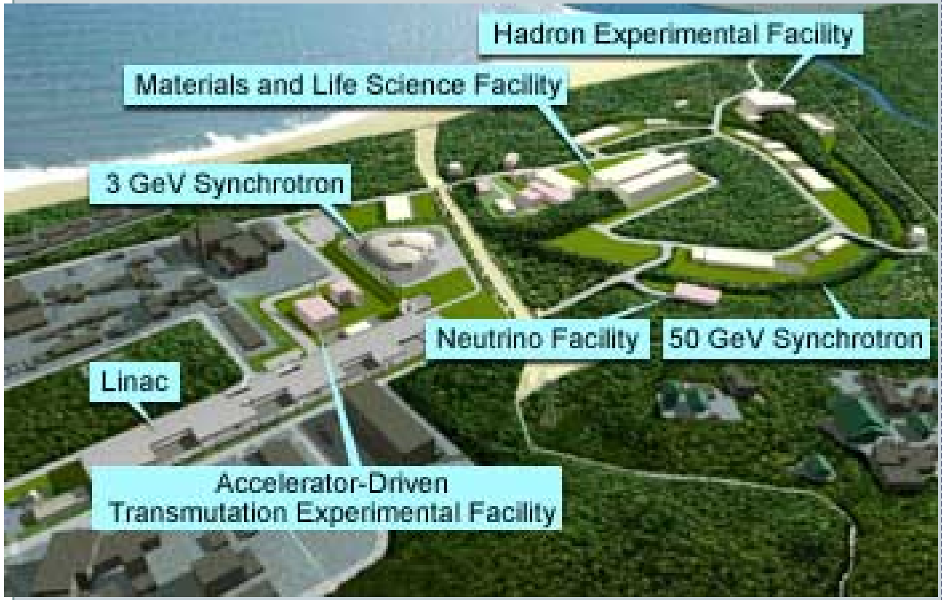
The XAL application development environment has been installed as a part of the control system for the Japan Proton Accelerator Research Center (J-PARC). XAL was initially developed at the Spallation Neutron Source (SNS) and has been described at length previously. Included in XAL is an online model for doing quick physics simulations. We outline the upgrades and enhancements to the XAL online model necessary for accurate simulation of the J-PARC linac and transport system.

Outline

1. Motivation/Background
2. Space Charge Verification
3. Probe Component Refactoring
4. Algorithm Component Refactoring
5. Element Refactoring/Additions
6. Conclusion

1. XAL: Moving beyond Version 1.0

J-PARC Facility



SNS Facility

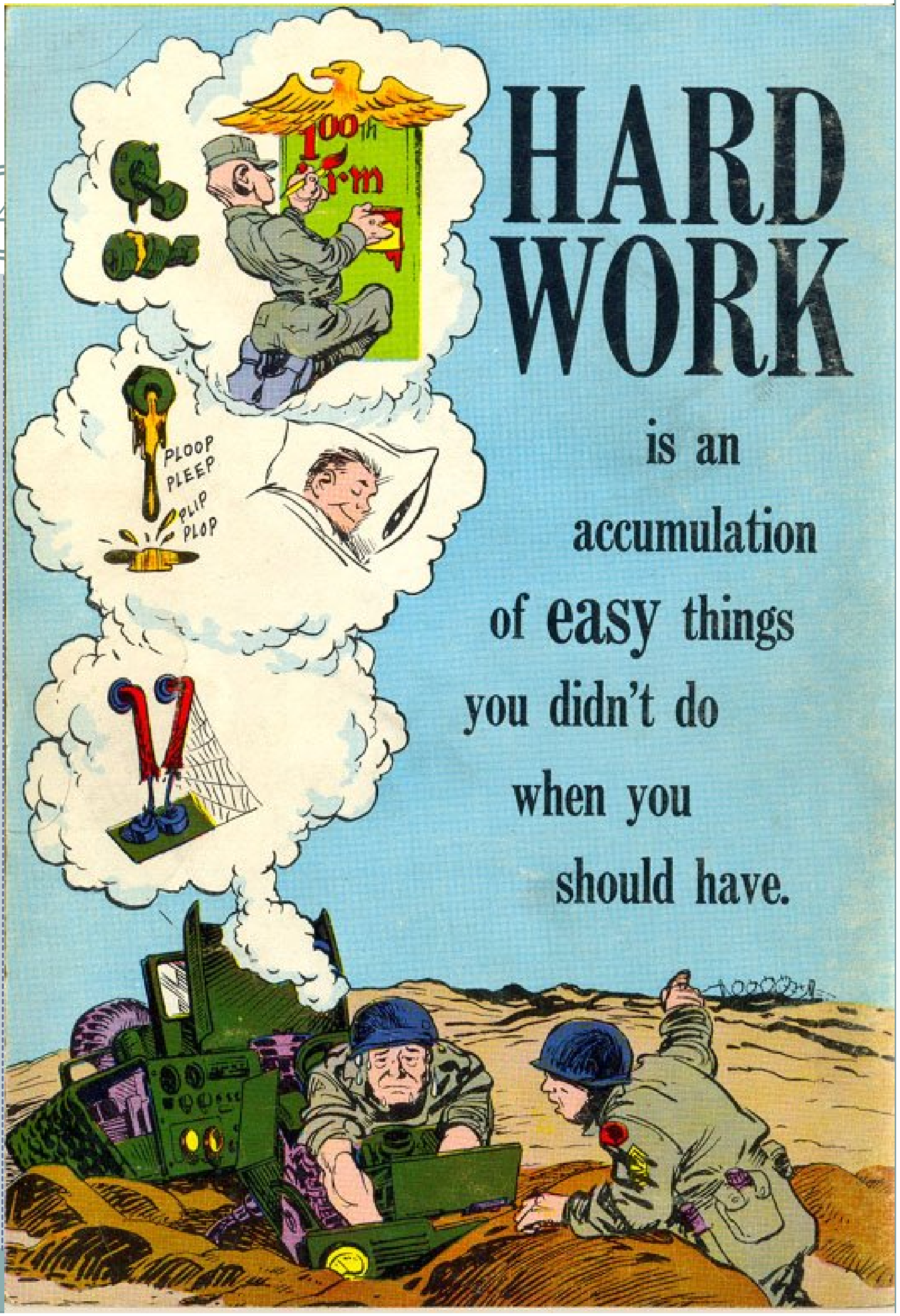
XAL is installed at both facilities (although modifications were necessary)
(Nice work Sako-san!)

1. Motivation

XAL was originally designed to be site-independent. However, it is difficult to consider “everything.” Also, SNS schedule pressures forced several site-specific “short-cuts.”

Modifications were necessary to generalize some aspects of XAL and to add features specific to J-PARC operation.

Many new features were added to the online model since its inception. The implementation was becoming brittle.

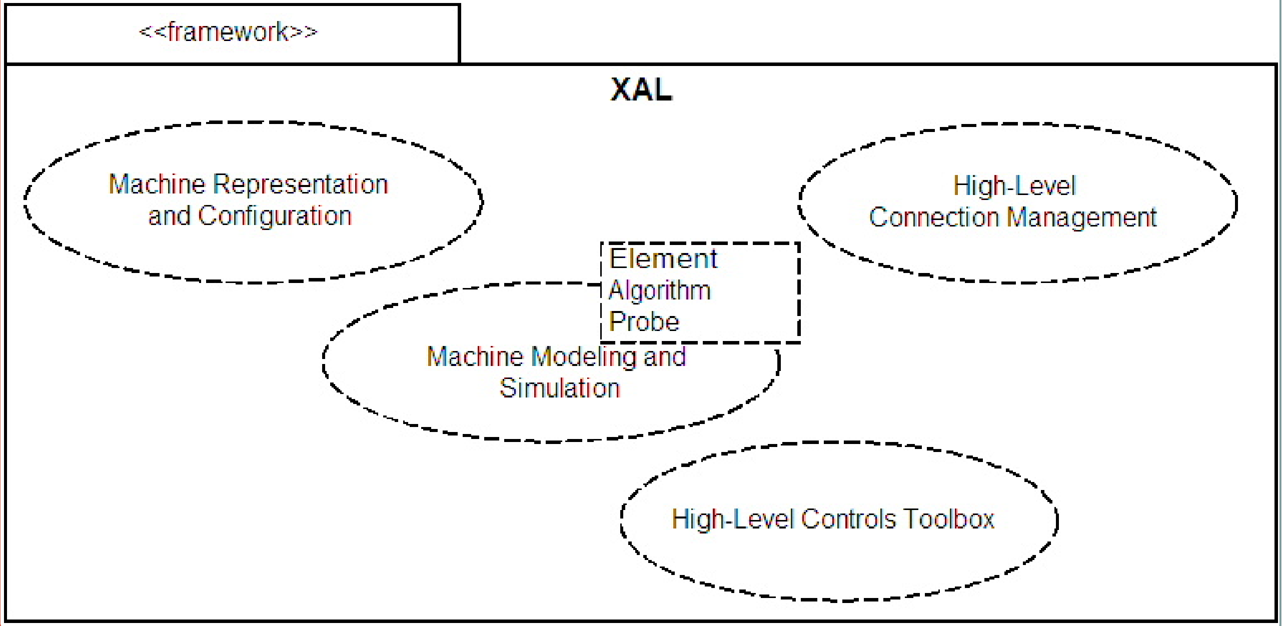


1. Background: XAL Architecture

XAL was designed with the following objectives:

- ➢High-level connection management
- ➢ Hware representation (introspective)
- ➢Tool suite
- ➢ Fast simulation (online model)

Here we consider modifications to the *model component* of XAL

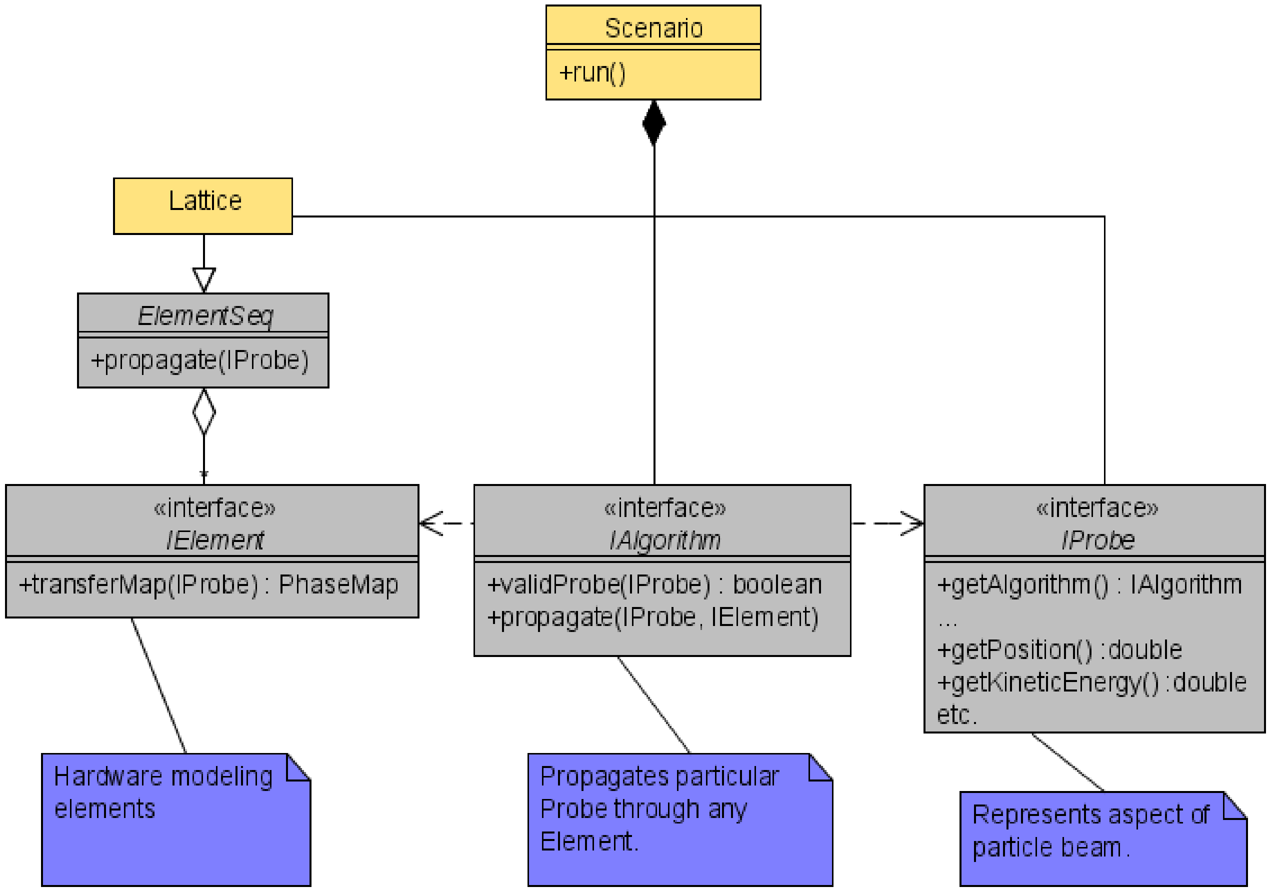


Element Algorithm Probe

XAL Online Model implementation based upon Malitsky's **Element/Algorithm/Probe** design pattern.

The online model is divided into three corresponding software components, all encapsulated with the `Scenario` class.

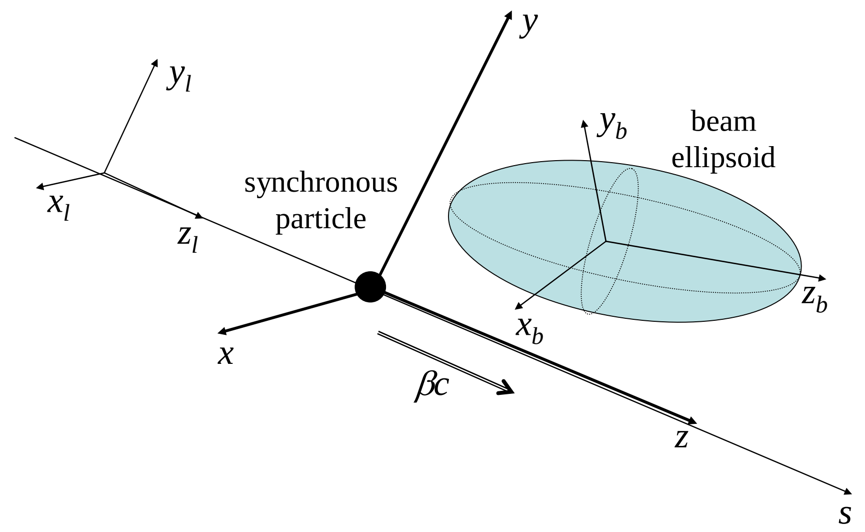
1. Online Model Architecture



2. Space Charge Effects Verification

After exhaustive verification process online model and Trace3D show exact agreement

- Located, coalesced, and corrected physical and mathematical constants in Trace3D
- XAL steps exactly like Trace3D
  - Previously both used methods accurate to $O(h^3)$, but errors accumulate and affect simulation
- Fixed several bugs in online model
  - Off-centered envelope tracking error
  - Lorentz-transform error
  - Tilted-beam error



SPACE CHARGE ALGORITHM

- Lorentz transform to beam frame
- Geometric transform from synchronous frame to ellipsoid “natural coordinate frame”
- Apply electric field kick (elliptic integrals)
- Inverse transforms

3. Probe Component: Original

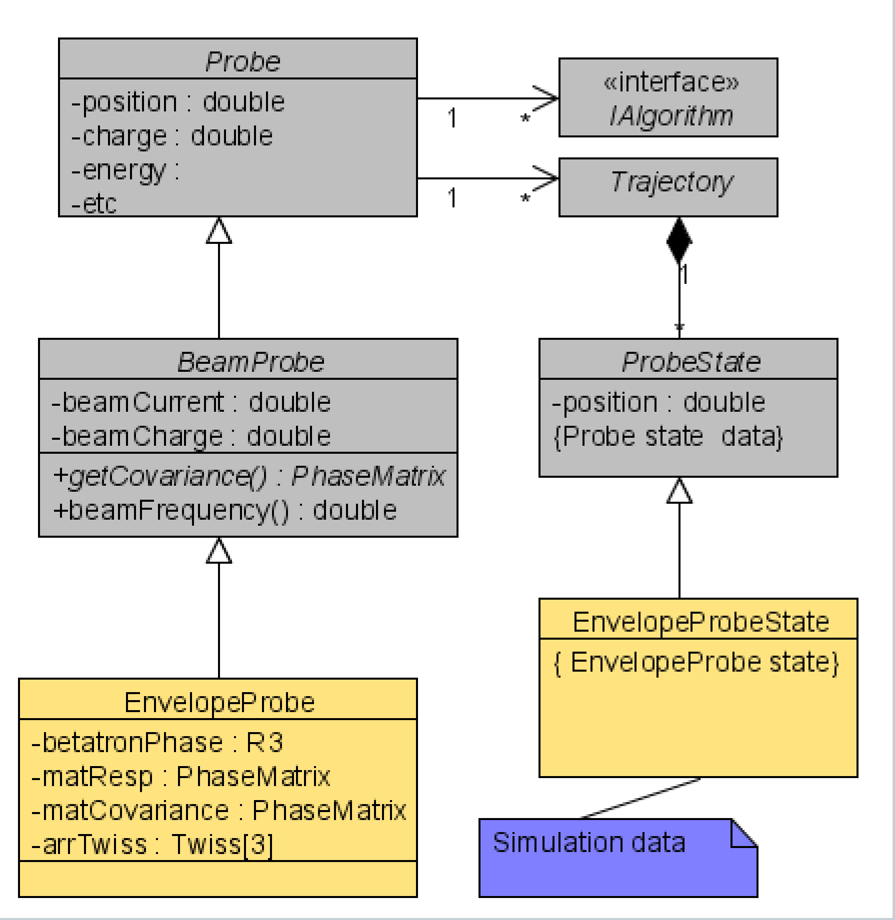
Recall`Probe` objects model aspects of beam (e.g., centroid, envelope, etc.)

`BeamProbe` hierarchy became dangerously inconsistent

- Beam current and beam charge were fundamental attributes
  - Frequency could be computed

- `EnvelopeProbe` contained redundant state information
  - Covariance matrix (2nd moments)
  - Twiss parameters

- Covariance generalizes Twiss parameters
- Two state variables had separate dynamics
- Heavy-weight simulation data

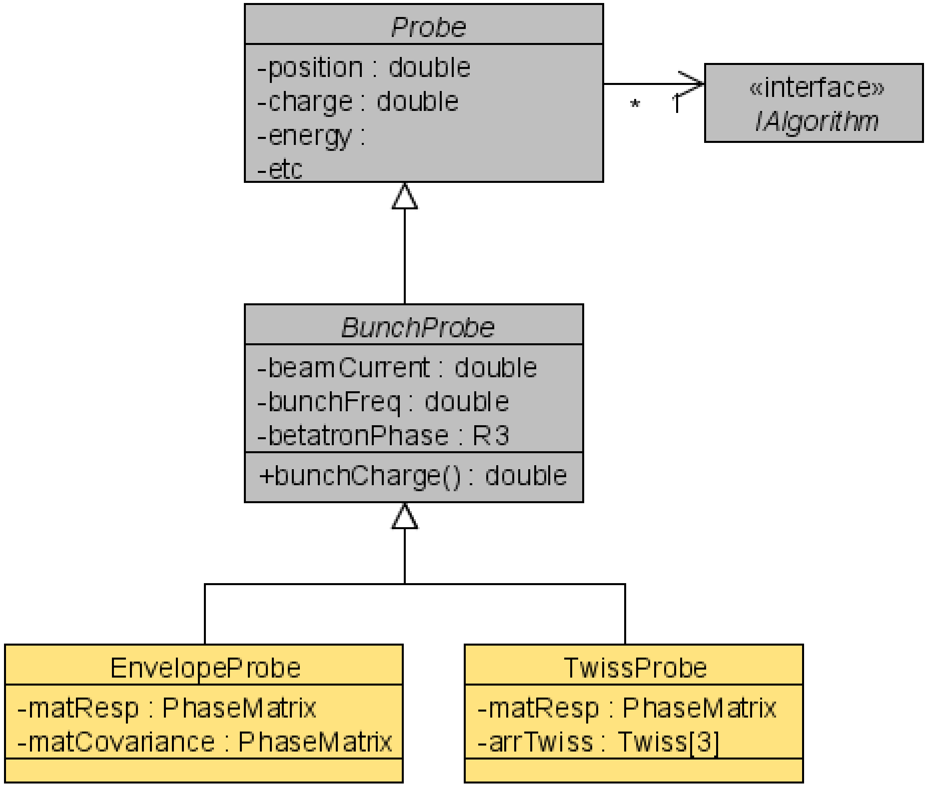


3. Probe Component: Refactored

`BeamProbe` hierarchy refactored

- Renamed `BunchProbe` to reflect changes
- Beam current, bunch frequency fundamental attributes
- Twiss parameters given separate Probe class `TwissProbe`

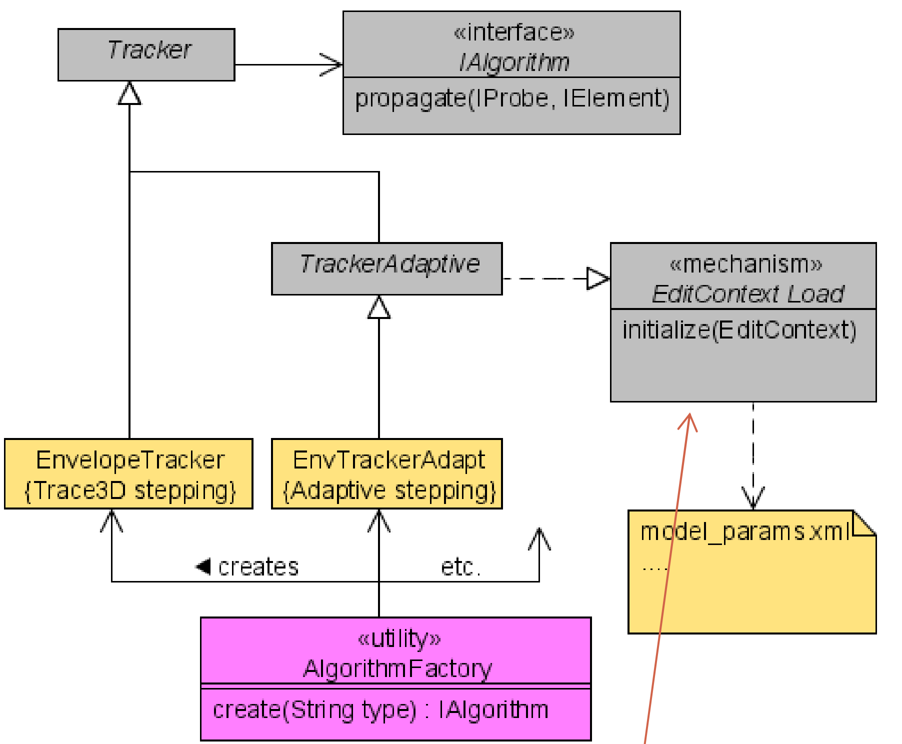
- `TwissProbe`
  - Lightweight variant of `EnvelopeProbe`
  - Ignores phase coupling



4. Algorithm Component

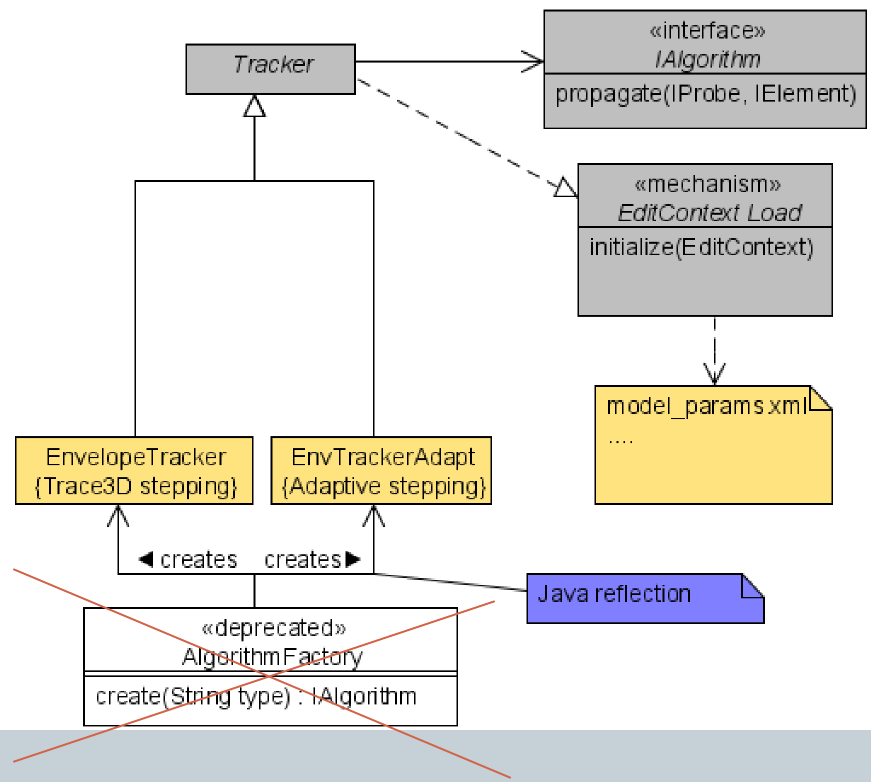
(Recall Algorithm objects propagate Probes through Elements)

Original



Automated probe initialization

Refactored



5. Bending Dipole Element

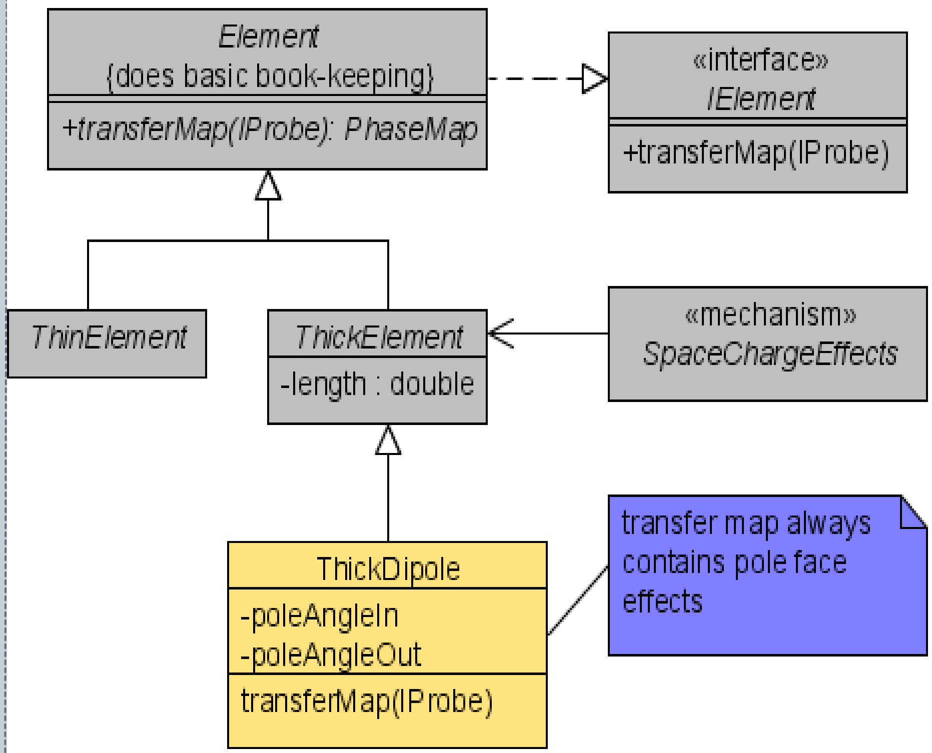
There were originally two implementations of a bending dipole

- `ThickDipole`
  - Modeled design trajectory effects
  - No space charge effects
    (did not conform to architecture)
- `IdealMagWedgeDipole`
  - No design trajectory effects
  - Facilitated space charge effects

The problem occurs when modeling pole face effects

- Make bending dipole a composite element

`ThickDipole` and Space Charge Mechansim

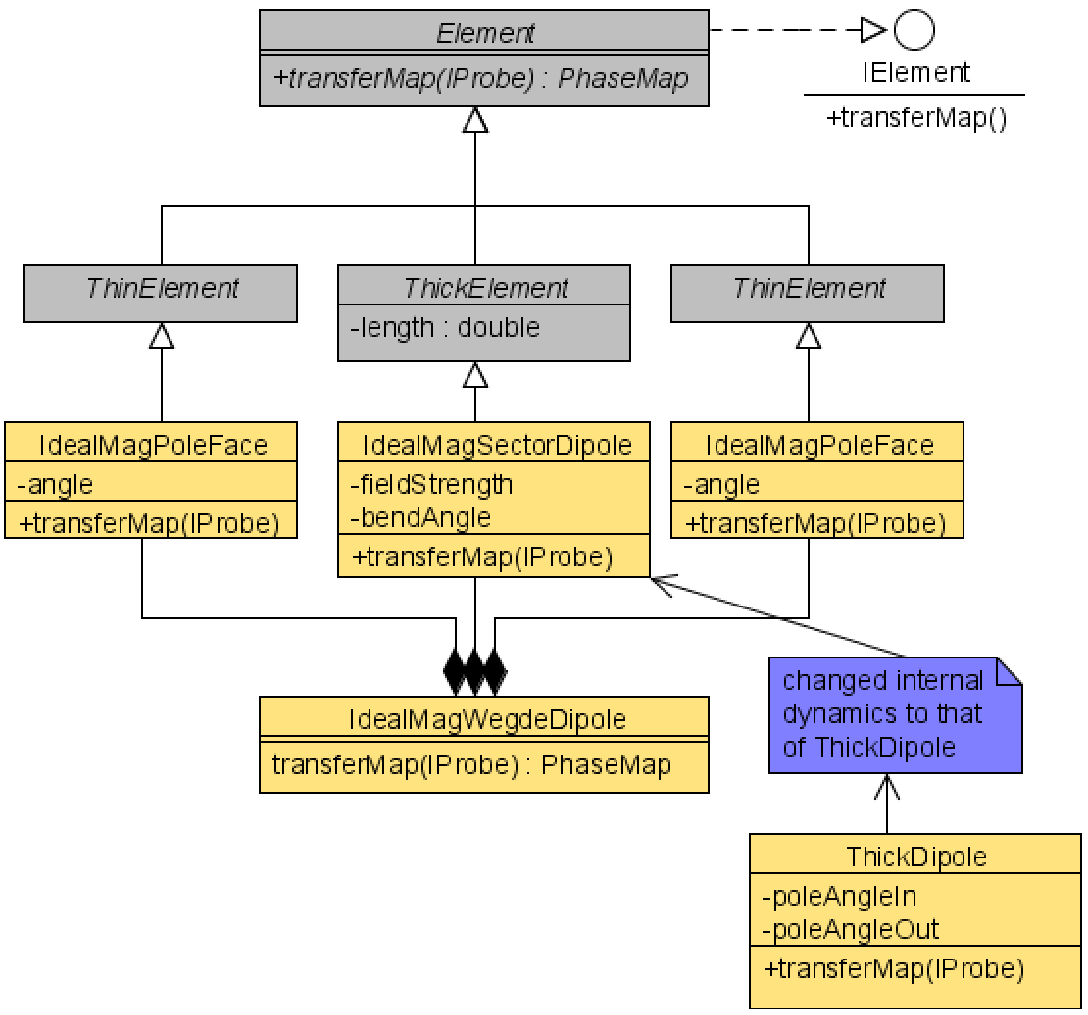


5. Bending Dipole Element Refactored

`IdealMagWedgeDipole`

- Composite element consists of
  - Two pole face as thin lenses
  - One magnet body that is space charge compliant

Moved design path dynamics of `ThickDipole` into magnet body of composite element



5. RF Gap Modeling Element

With an RF gap, emittance can increase from a finite longitudinal phase spread $\Delta\phi$

- This mechanism was added to the XAL online model
- Implemented in `Algorithm` component for `EnvelopeProbe`
- Implemented the same model as Trace3D

It was discovered that the Trace3D model is invalid in the longitudinal direction

- Trace3D uses an approximation accurate only for $\Delta\phi \ll$
- Emittance growth blows up as $\Delta\phi \rightarrow \infty$

5. RF Gap Modeling Element

Emittance increase can be represented as an additive term $\Delta\varepsilon$

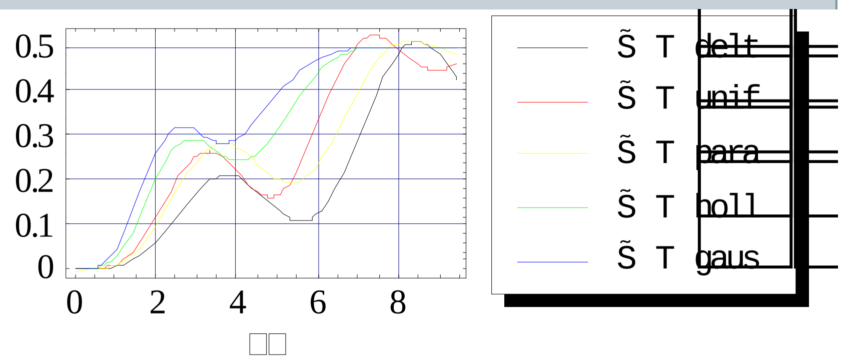
$$\Delta\varepsilon^2 \propto \beta^2 G(\phi_s, \Delta\phi)$$

where ϕ_s is synchronous phase and *growth function G* has form

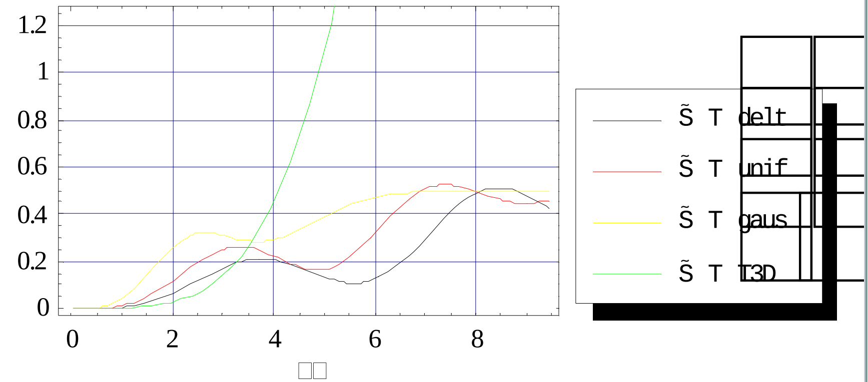
$$G(\phi_s, \Delta\phi) = S(\Delta\phi) - T(\Delta\phi)\sin^2\phi_s$$

- T and S are bounded functions with limits of ½ and 0, respectively
- Plot $T(\Delta\phi) - S(\Delta\phi)$ to see worst-case emittance growth

$\Rightarrow$ Trace3D emittance growth inaccurate for long (debunched) beams



Longitudinal $G(90, \Delta\phi)$ for different distributions

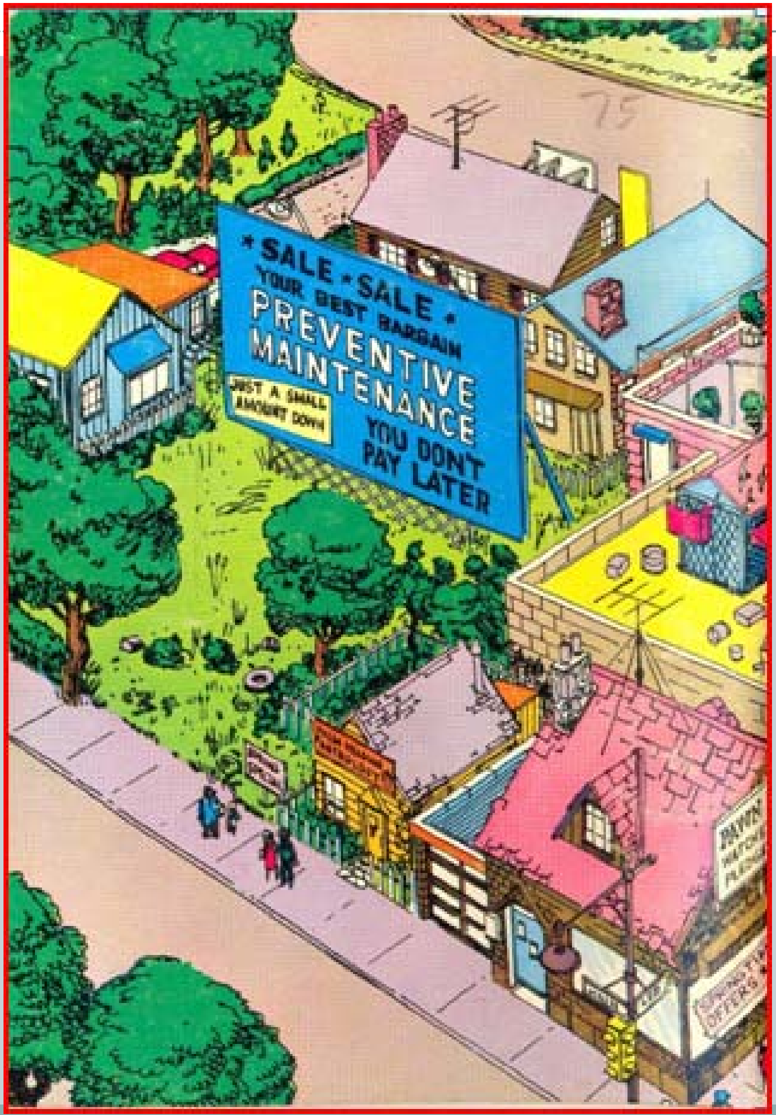


Longitudinal $G(90, \Delta\phi)$ including Trace3D

6. Conclusion

Many new features have been added to the online model since its original design, both at SNS and at J-PARC. The overall result of this work was not only the addition of new capabilities, but importantly re-engineering of the software to accommodate these mechanisms in a framework that is

- Robust
- Understandable
- Maintainable




Issue 125
PS
1963 Series
THE
PREVENTIVE
MAINTENANCE
MONTHLY
SALE ★ SALE
YOUR BEST BARGAIN
PREVENTIVE
MAINTENANCE
JUST A SMALL
AMOUNT DOWN
YOU DON'T
PAY LATER
SPRING
SALE
SPECIALS FOR SOLDIERS
ARMY NA
STORE
UNIFORMS MAD
SPRING SPECIAL
SPECIAL
RATES
Fancy
Rooms
etc.
FORT PINECONE
PAWN BROKERS
WATCHES
PLEDGES
PALM READING
ASTROLOGY
SPRING
SPECIAL
SPRINGTIME
OFFERS
SALE
SPRING
MAIN
LEE
Will
Eisner

BeamProbe Hierarchy

The *Shock and Awe* approach to interface design

- Obfuscating form and function
- Write code nobody can read
- Make it brittle so nobody will touch it
- Create dangerous situations which explode only when it really counts

Question: what do you get when you call `BeamProbe.getTwiss()` ?

Moral ⇒ Refactoring is good

- It may not be sexy but it will save a lot of future time and anguish

BeamProbe

moments() : PhaseMatrix //2nd moments

twiss () {moments(). compTwiss() }

getTwiss() { return twiss() }

EnvelopeProbe

covariance : PhaseMatrix // 2nd moments

twiss : Twiss[3] // Twiss parameters

---

moments() : { return covariance }

twiss() : { covariance.compTwiss() }

getTwiss() : { return twiss }