SiLab Gantry Software Development

Caleb Fangmeier Jose Monroy

University of Nebraska - Lincoln

CMS FPIX Meeting - April 27, 2013

Outline

Timeline

- Spring 2012 - UNL gantry acquired

Timeline

- Spring 2012 - UNL gantry acquired
- July 3, 2014 - Initial commit of gantry software. Gluing routine already heavily under development.

Timeline

- Spring 2012 - UNL gantry acquired
- July 3, 2014 - Initial commit of gantry software. Gluing routine already heavily under development.
- Summer 2015 - Phase I pre-production begins. Software begins to be stressed.

Timeline

- Spring 2012 - UNL gantry acquired
- July 3, 2014 - Initial commit of gantry software. Gluing routine already heavily under development.
- Summer 2015 - Phase I pre-production begins. Software begins to be stressed.
- Fall 2015 - Identify memory leak in potting routine Limits potting to 1 module per session $\rightarrow \approx$one hour per module

Timeline

- Spring 2012 - UNL gantry acquired
- July 3, 2014 - Initial commit of gantry software. Gluing routine already heavily under development.
- Summer 2015 - Phase I pre-production begins. Software begins to be stressed.
- Fall 2015 - Identify memory leak in potting routine Limits potting to 1 module per session $\rightarrow \approx$one hour per module
- Oct. 14, 2015 - Begin software rewrite. Focus on potting.

Timeline

- Spring 2012 - UNL gantry acquired
- July 3, 2014 - Initial commit of gantry software. Gluing routine already heavily under development.
- Summer 2015 - Phase I pre-production begins. Software begins to be stressed.
- Fall 2015 - Identify memory leak in potting routine Limits potting to 1 module per session $\rightarrow \approx$one hour per module
- Oct. 14, 2015 - Begin software rewrite. Focus on potting.
- March 30, 2016 - New routine deployed.

Original Software

- Performed needed actions and served as a good R&D platform
- But...
  - Unreliable - Memory leak results in unpredictable crashes.
  - Unmaintainable - Essentially a single script that has grown to do the whole routine.
- Clearly a rewrite is in order.

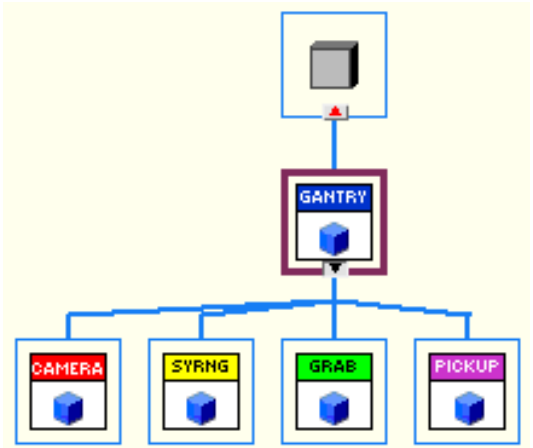
Outline

The Solution

- Qualities of old software made it impractical to reuse parts.
- Rewrite software from scratch!
- Bring in a UNL `LabVIEW` expert, Dustin Dam, to consult with redesign.

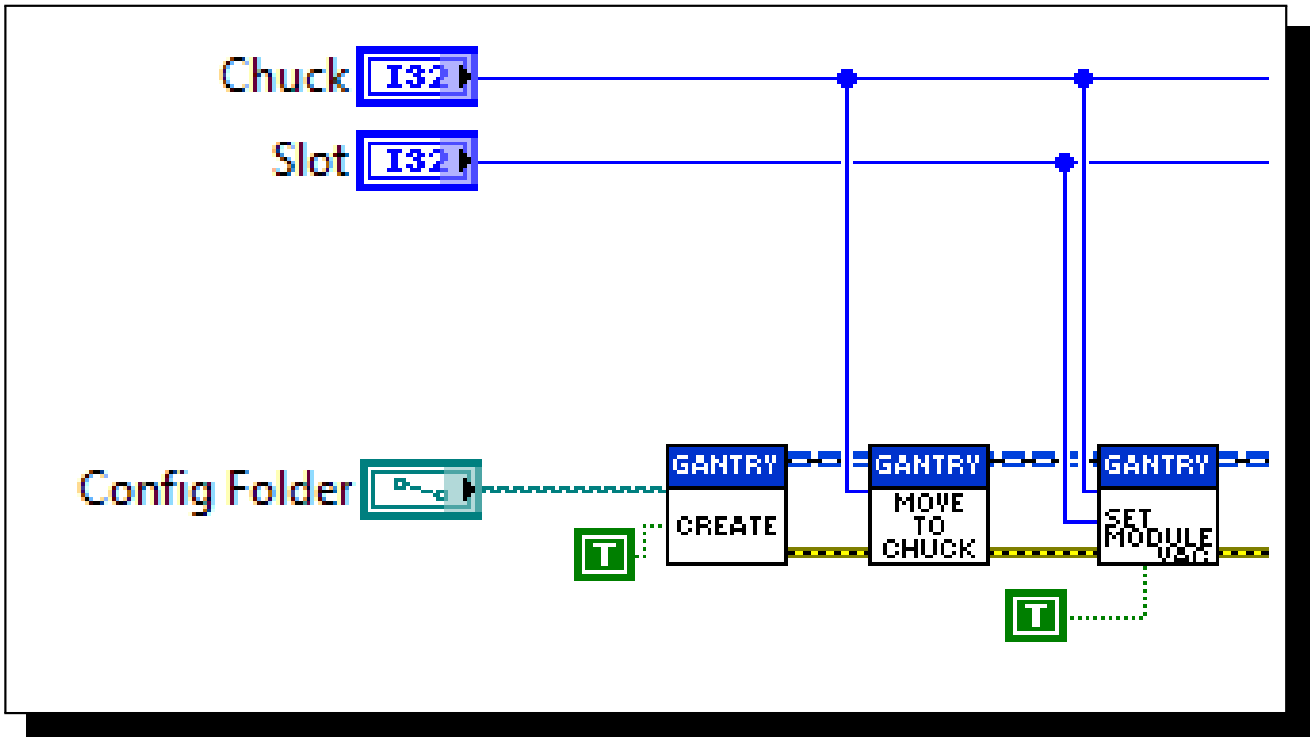
OO Design

- Adhere to `LabVIEW` software best practices.
  - Don't fight data-flow.
  - Small functions.
  - Organize code into `LabVIEW` project.
- Take advantage of `LabVIEW`'s Object-Oriented features.
- Encapsulate *all* hardware interaction within a single class (`Gantry`).
- Use inheritance to model specialized functionality of the gantry's "Tools".

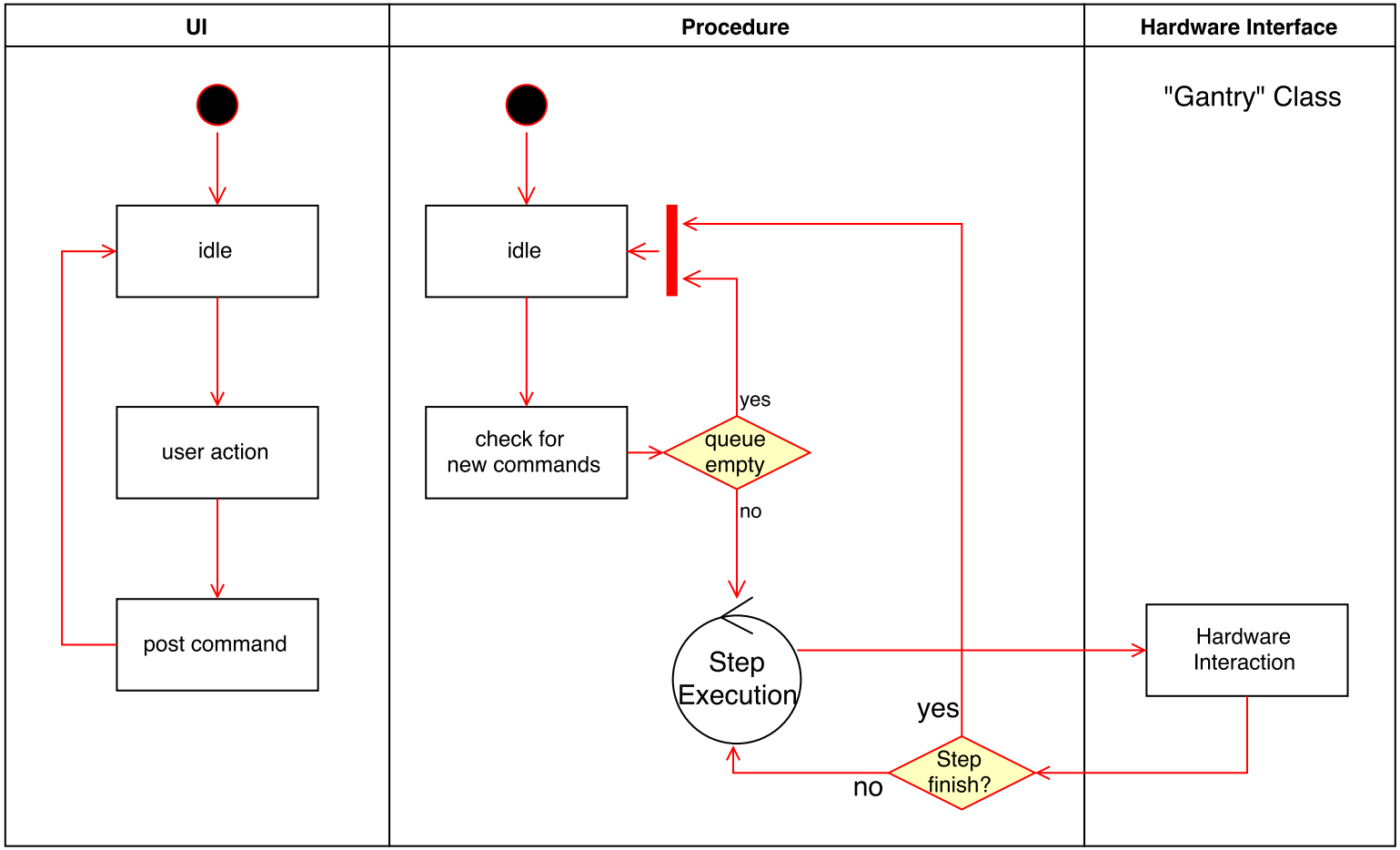


:

Code Sample

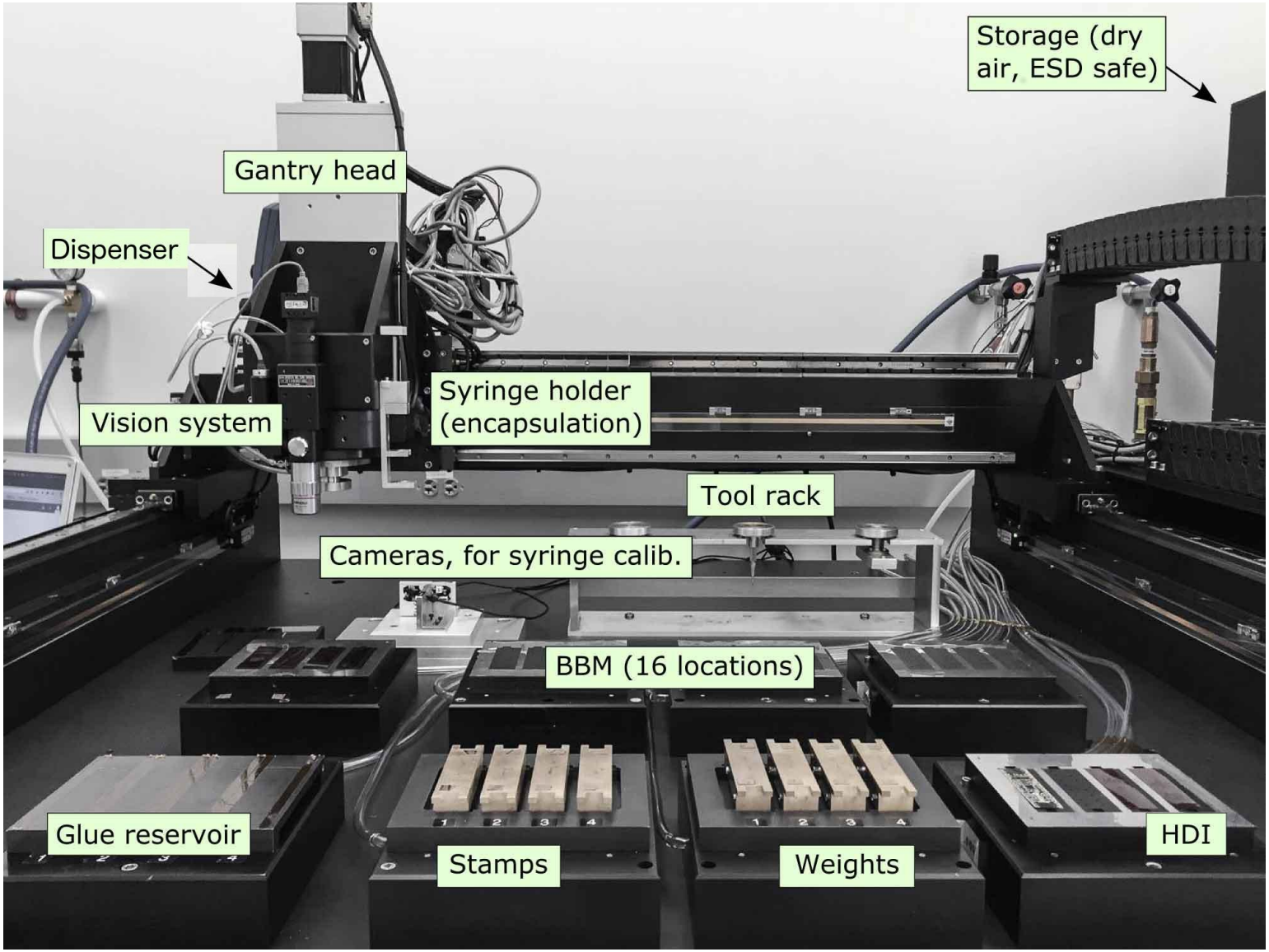


Encapsulation Routine Architecture

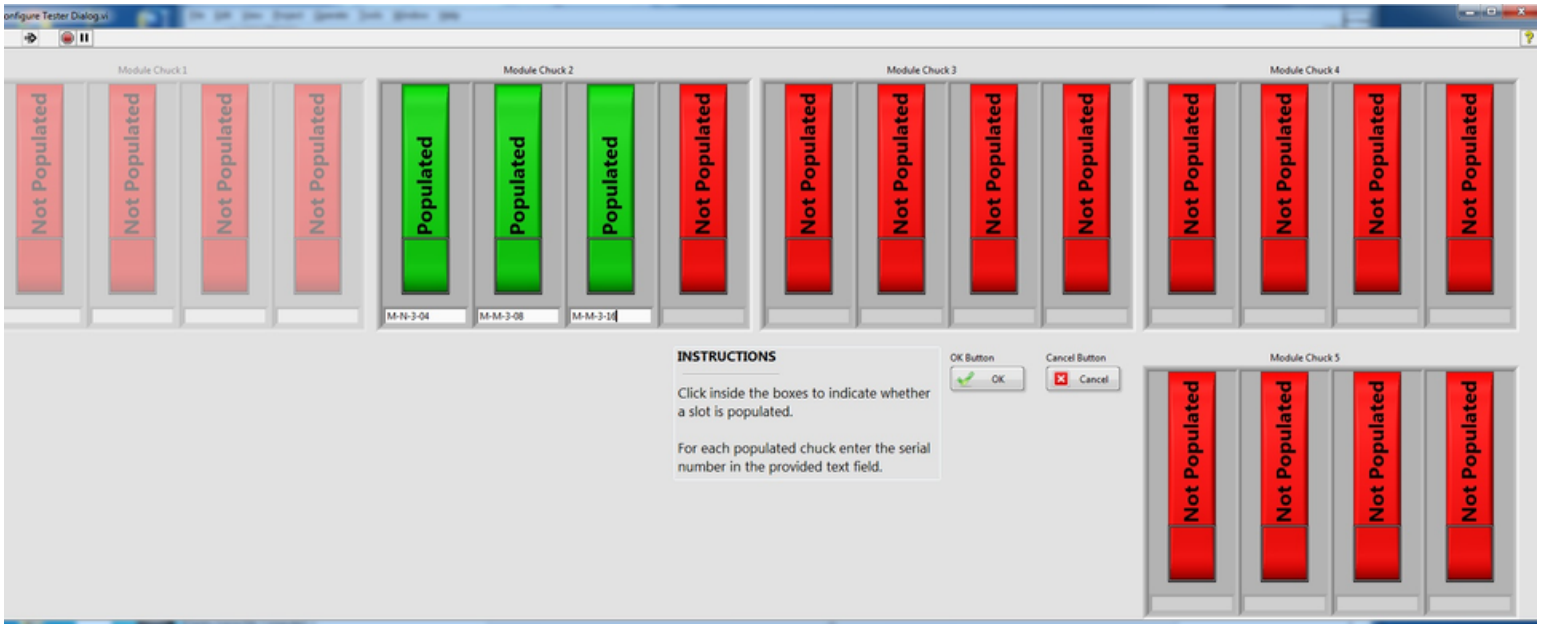


Outline

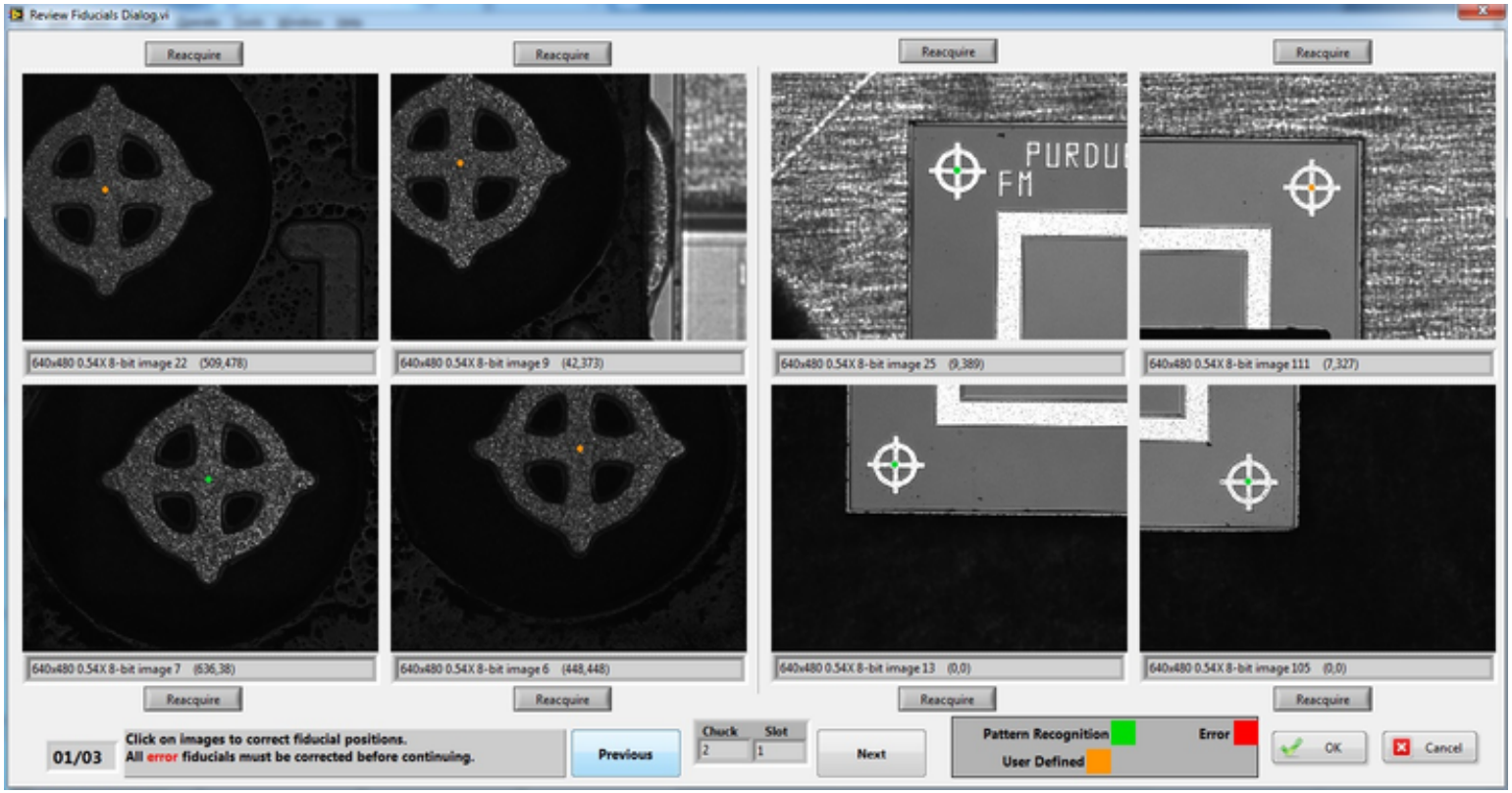
Gantry Table Setup



Session Configuration

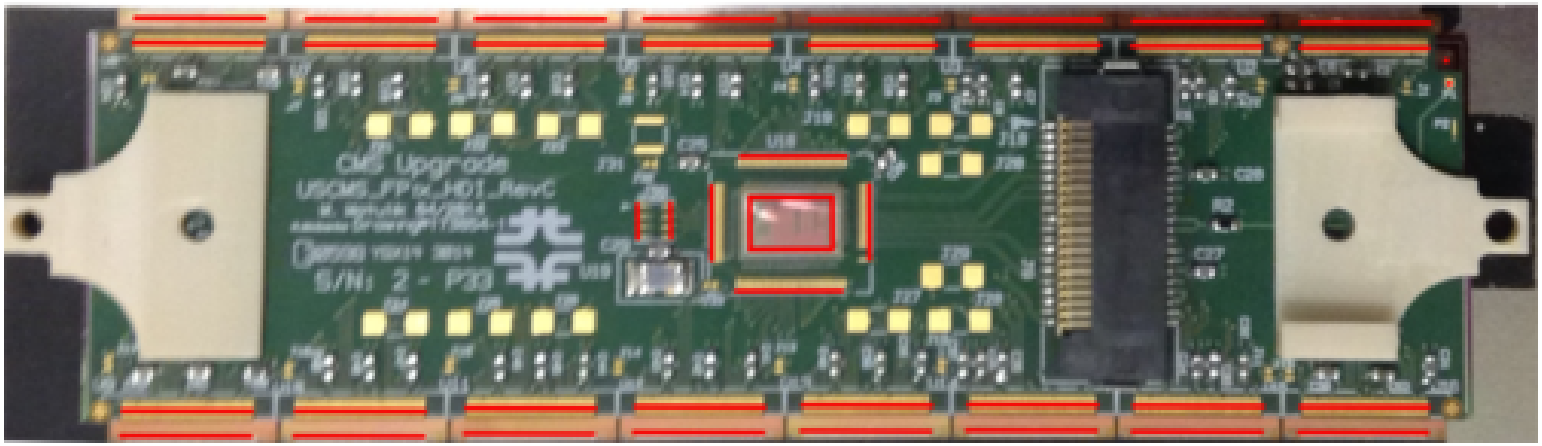


Fiducial Verification



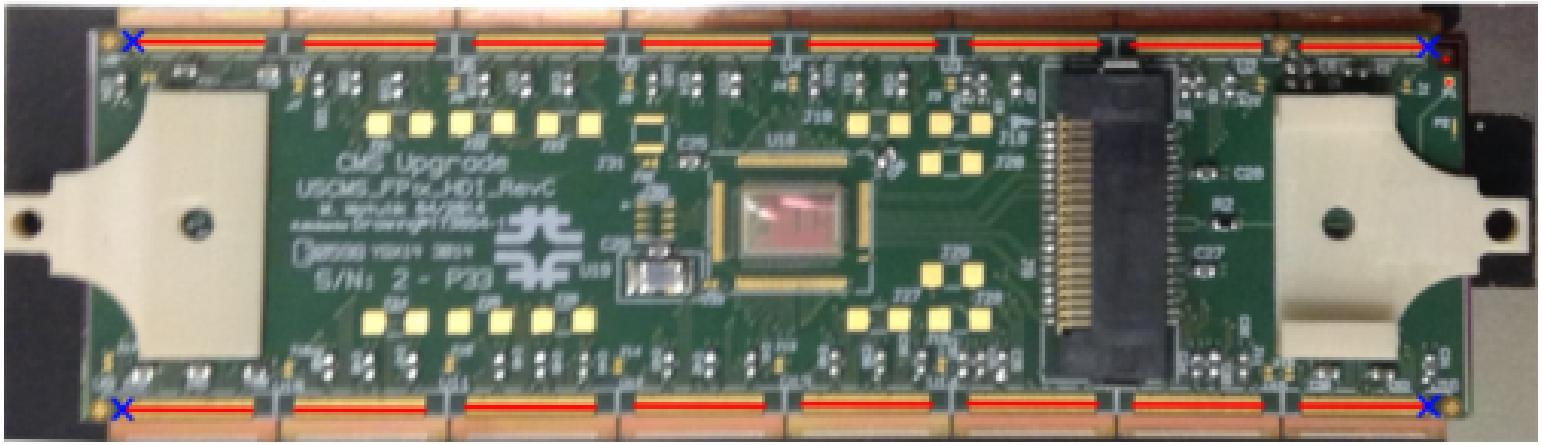
Potting Locations

- Red lines indicate what must be potted.
  - HDI$\longleftrightarrow$ROC Bonds
  - TBM$\longleftrightarrow$HDI Bonds
  - Address Pad Bonds
  - HV Bonds

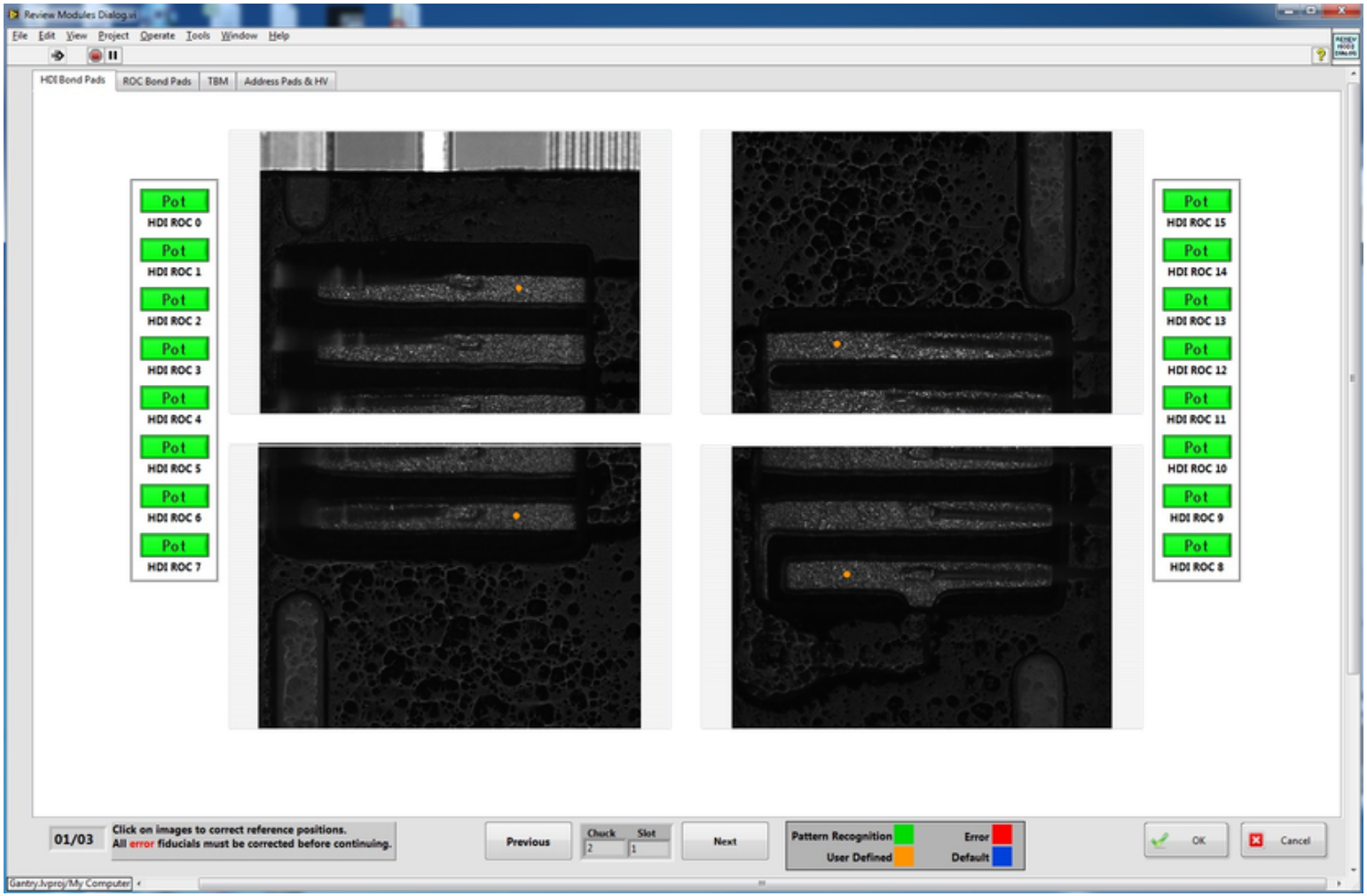


Potting Locations - HDI Bond Pad Groups

- Take images at blue crosses.
- User selects point within images.
- Points are used to calculate the red line segments.



UI - HDI Bond Pad Groups



Outline

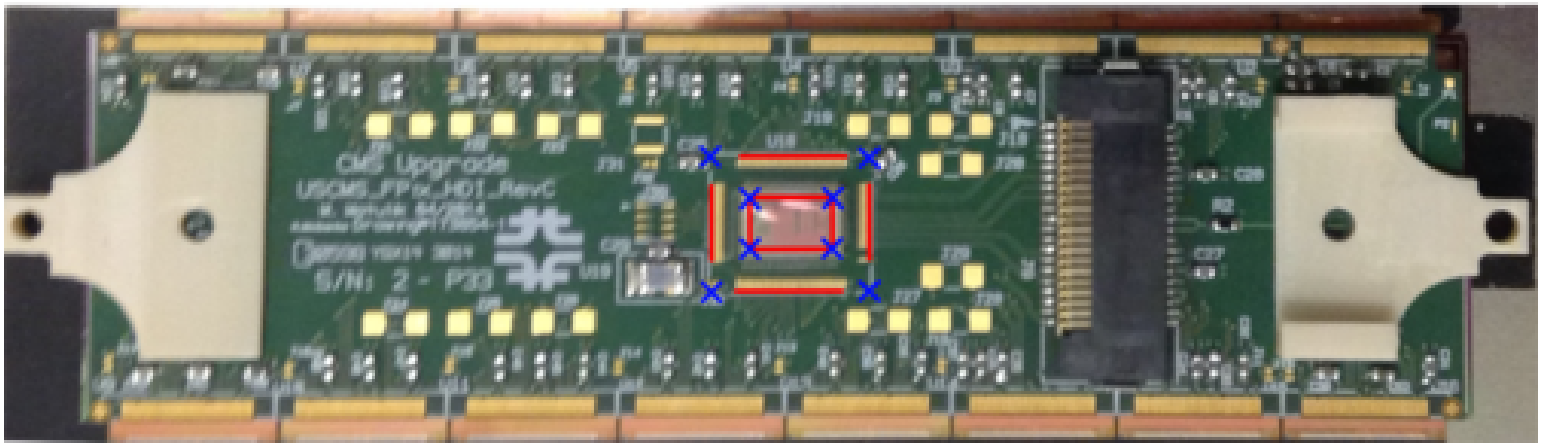
Summary

- Potting routine deployed March 30
  - Slow ramp up, but we can now encapsulate 8 modules per session.
  - 16 minutes per module (down from an hour with original routine)
  - 76 modules encapsulated without incident.
- Code can be found here (Requires UNL credentials)
- Potting progress and basic analysis of performance can be found here.

Outlook

- Will rewrite gluing routine over summer with new codebase.
- Can look into alternative code hosting if others want to collaborate.

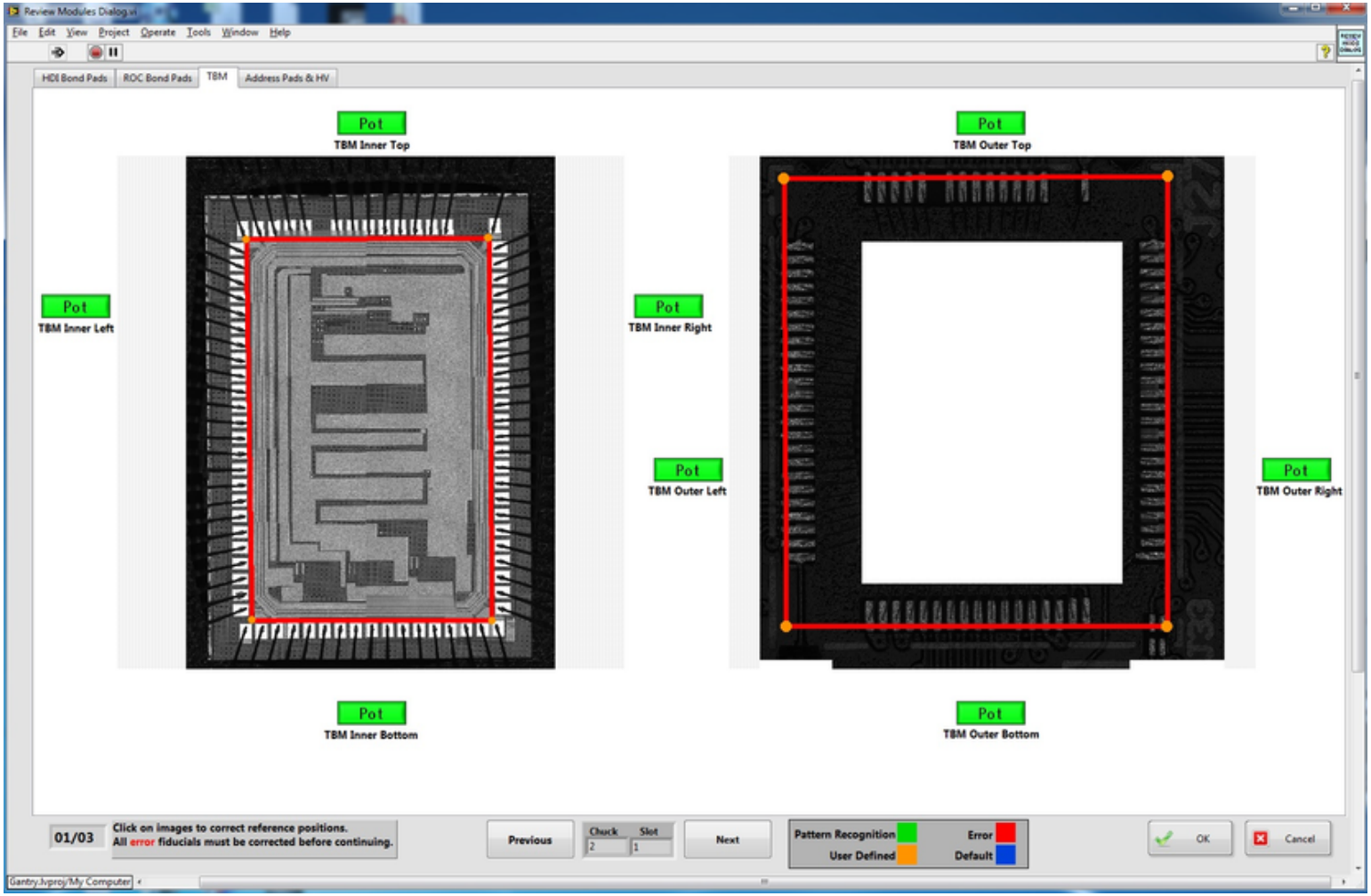
Potting Locations - TBM



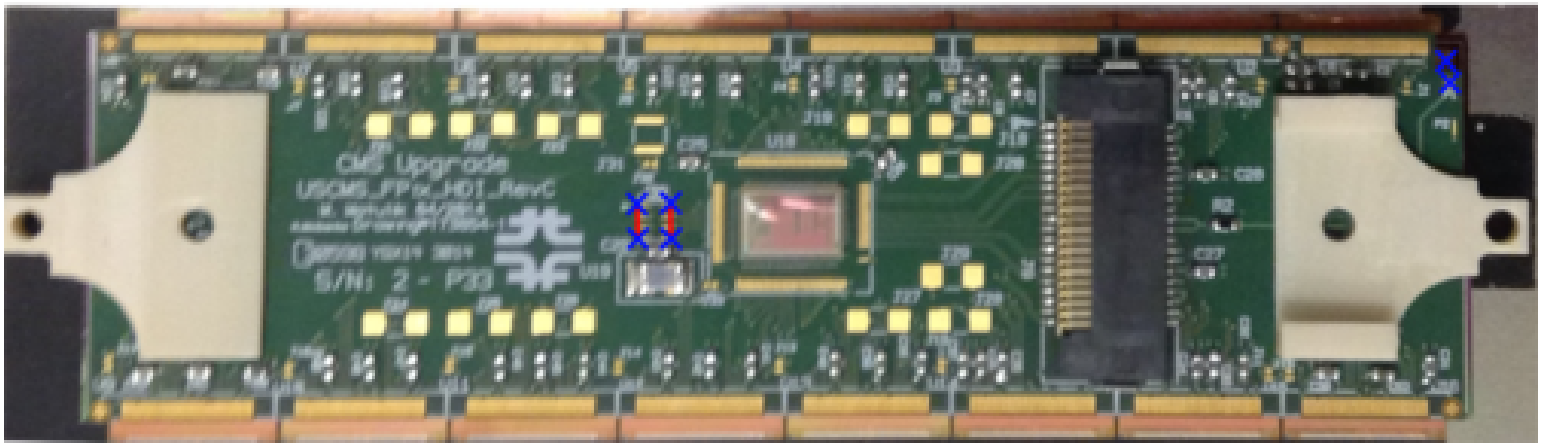
UI - TBM

SiLab Gantry Software Development

Caleb Fangmeier, Jose Monroy



Potting Locations - HV & Address Pads



UI - HV & Address Pads