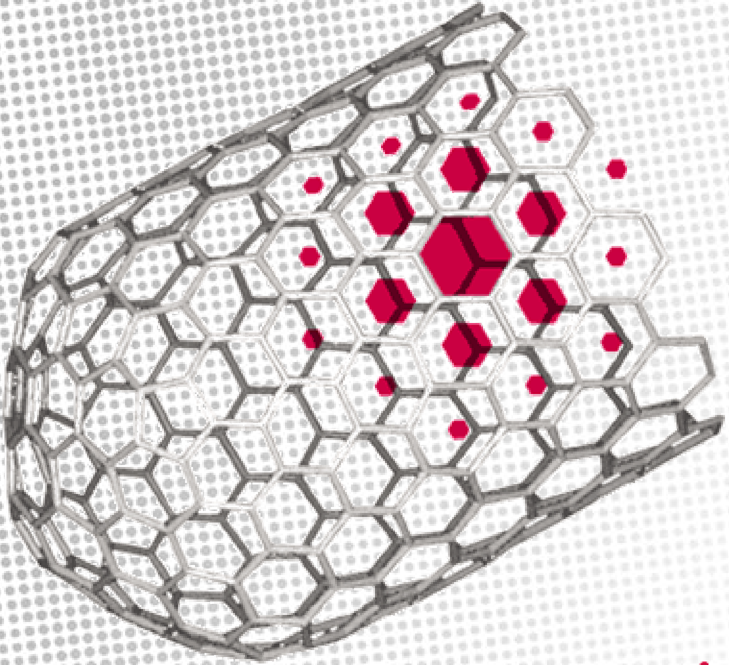
Graceful Degradation

June 4, 2007

Michael Leichsenring



FEI COMPANY™

TOOLS FOR NANOTECH

Presentation Outline

- Introduction
- System and Software Architecture
- Why Needed?
- Our Solution
- Required Changes
- However

Introduction

Developer of Electron Microscopes

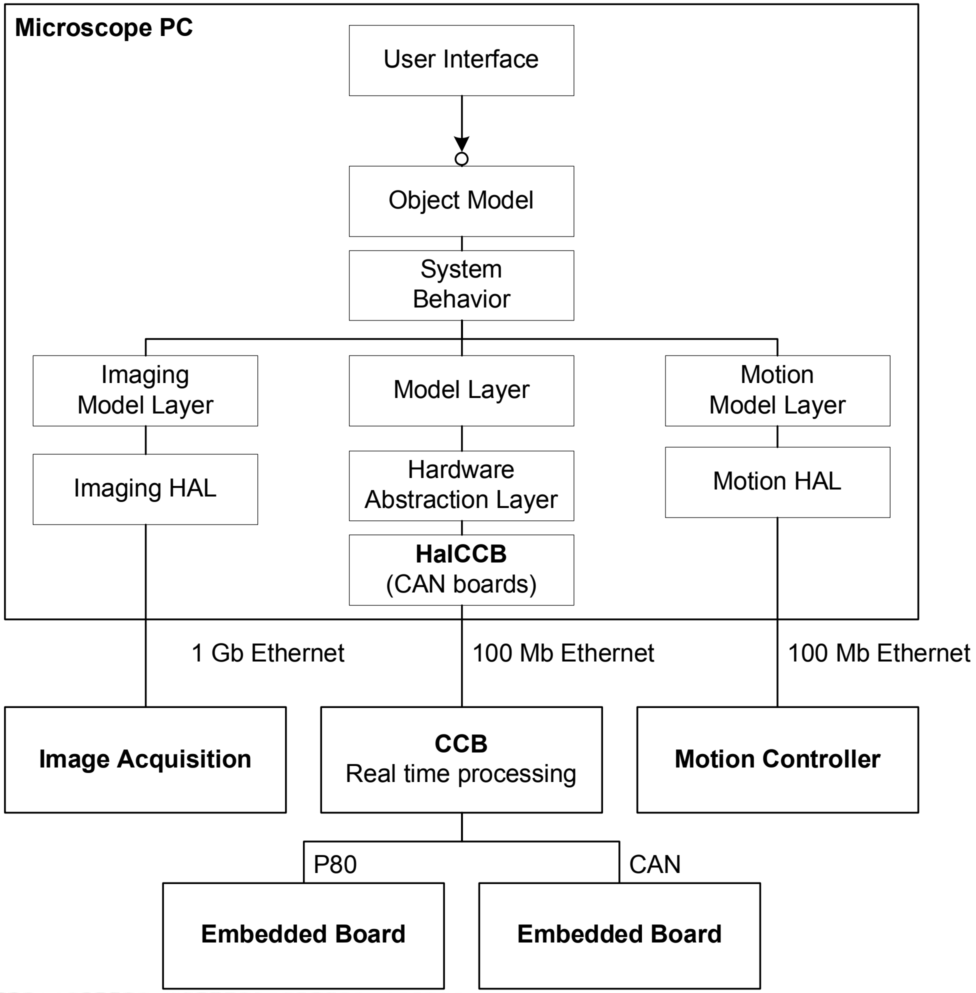
Main product lines:

- Transmission Electron Microscope (TEM)
- Scanning Electron Microscope (SEM)
- Focused Ion Beam (FIB)
- Dual Beam (DB = SEM + FIB)

Each product line supports large number of configurations

Configuration combination of hardware and software

Instrument Architecture



Software Architecture

- Based on COM components, called Bricks
- Startup/Shutdown handled using lifecycle phases
- Embedded board connections managed by HAL
- Model Layer understands board behavior

Why we need Graceful Degradation

Customer Site:

- Detecting and reporting hardware problems
- Minimize service time by supporting hot swapping

Manufacturing:

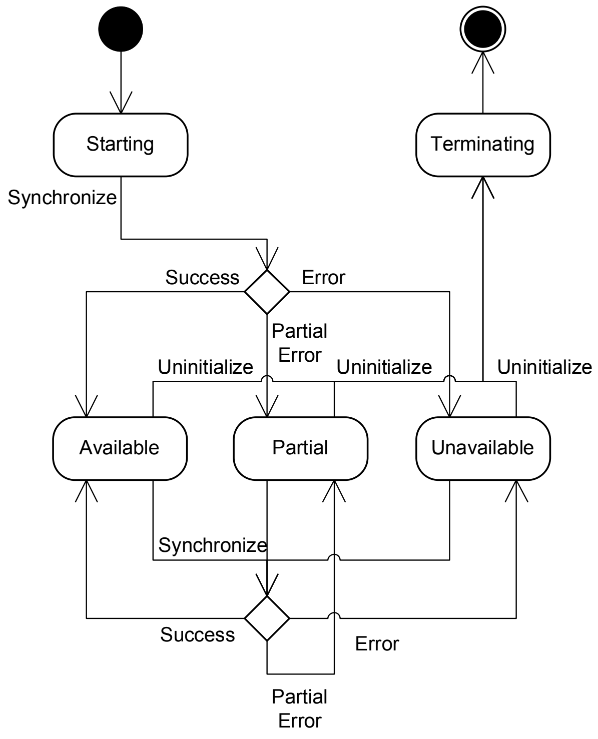
- Support lean manufacturing (assembling modules)
- Simplify configuration management

Actually we need: System Recovery

FEI™

Our Infrastructure Solution

- Generic state machine added to each component
- State change propagates through system
- Notification behavior handled by infrastructure (allows for change)
- Notification triggers actual “recovery”
- Component implements Synchronize method to support recovery
- Service tool to monitor states



Component Implementation

Infrastructure contains methods to switch state, however component functionality decides when. Infrastructure performs administration and propagation.

To support graceful degradation and recovery:

- Implement the Synchronize method with connected detection
- Implement disconnected detection (on each call when possible)

Usage state to:

- Determine and report available functionality
- Eliminate communication timeouts

FEI™

Required Changes

Infrastructure software:

- Add support for state machine
- Default behavior to support backward compatibility

Hardware Abstraction Layer:

- Rework connection handling (disconnect and connect)
- Detecting events “loss” due to short disconnects

Model Layer:

- Rework depends heavily on component
- Can be prioritized based on frequently occurring issues

However

- Concept already 4 years old
- A low priority project:
  Everybody wants it but without spending resources
- Not part of initial framework and guidelines, therefore lot of effort needed
- Open questions about instrument behavior
- Connection with component state model unclear