

Sustaining the National Ignition Facility (NIF) Integrated Computer Control System (ICCS) Over Its Thirty Year Lifespan

ICALEPCS 2017

October 2017

Barry Fishler
NIF ICCS Manager



NIF/ICCS will remain operational until at least Spring 2039

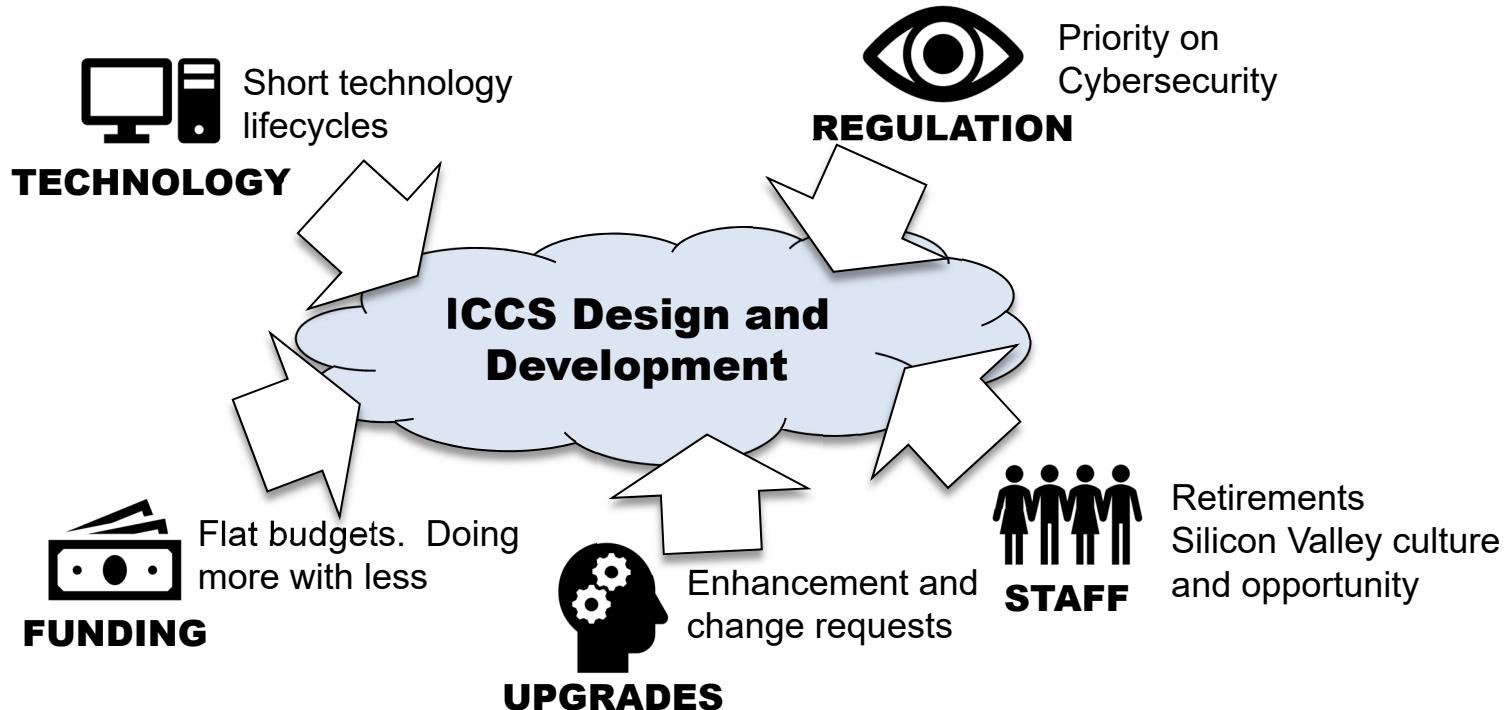
NIF was commissioned as an operational facility in 2009 with a 30 year planned lifespan

- ICCS coding started in the mid-1990s
 - Control system code has grown to over 3.5M lines
 - Much of that same code base is still in active use within NIF
 - Hardware has been used for 1000s of laser shots since operations commenced
- Multiple large influences are resulting in voluntary and involuntary control system updates
 - Demand from various programs for new capabilities
 - End-of-life for original software languages, hardware components, operating systems, and compilers
 - Operational need for high reliability, availability, and maintainability

Process and design changes are allowing us to manage this necessary volume of change

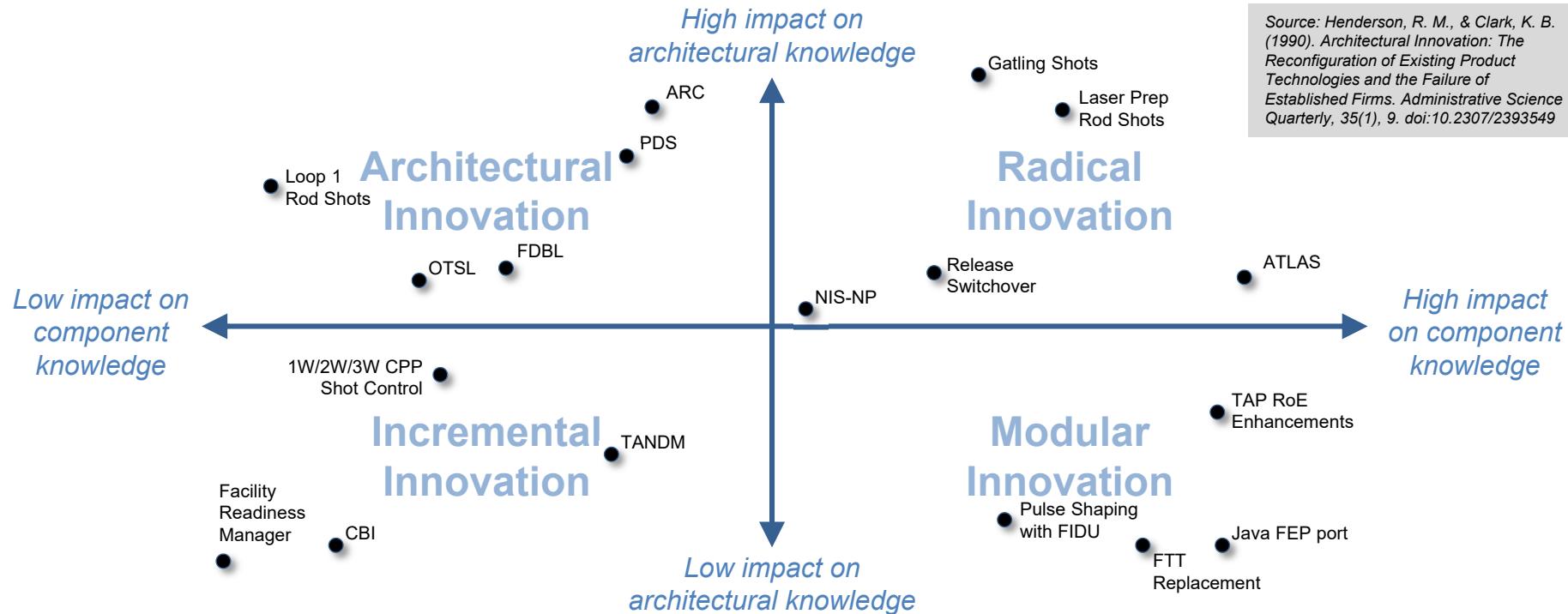
Influences on our design and development processes

Significant effort in managing project priorities with limited staff



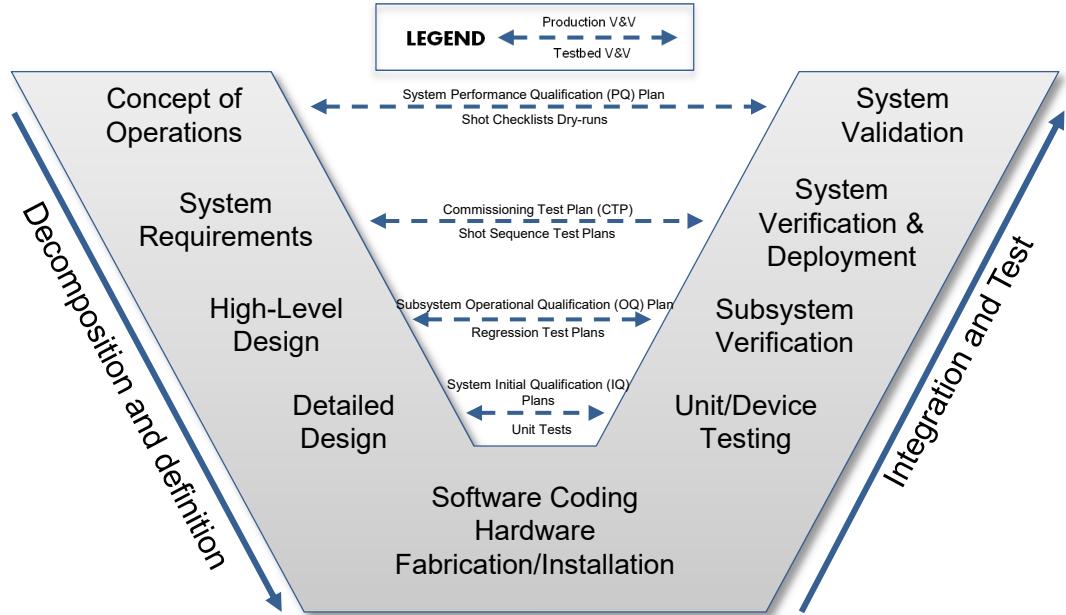
Development processes need to accommodate loss of experienced and knowledgeable developers

ICCS is undergoing significant architectural and component upgrades while NIF operates as a user facility



Separate strategies are required to address both architectural and component level changes while ensuring high RAM

Investment in specific areas of the V-model were necessary for long term sustainability



- Design documents and test plans fell out of date as they were not intrinsically updated in development process
- Could not rely on subject matter expert retention
- As the number of software capabilities grew, manual testing could not sustain quality

Lessons learned from initial Ada to Java porting attempts quickly identified areas for improvement in quality processes

Mandatory automated testing institutionalized as pre-ancestor for Java porting effort

Provides direct comparison of Ada vs Java results, including performance

- Too many test cases to qualify software in testbed solely through manual testing
- Schedule relief provided for porting milestones to focus on test frameworks and test case development
 - Substantial improvement on our ability to successfully deliver on those milestones
- Test case annotations:
 - “fast” for inclusion in continuous integration environment, and “slow” tests for running on-demand (e.g., by developer during coding)
 - Emulation, real hardware, or both
- Long term benefits:
 - Immediate qualification on component level changes, including hardware
 - Test cases document expected system behavior to facilitate more rapid staff onboarding

Test Case	Ada Status	Java Time (s)	Ada Status	Java Time (s)
<i>Common</i>				
checkEmulationStatus	PASS	3.002	PASS	2.999
checkHealthStatus	PASS	3.003	PASS	3.004
checkInterfaces	PASS	3.009	PASS	3.421
<i>MotorBasicsTest</i>				
checkSettings	PASS	3.031	PASS	3.422
getComponentStatus	PASS	3.033	PASS	3.011
isMovingWhileBusy	PASS	7.433	PASS	9.012
motorNotMovingWhileIdle	PASS	3.006	PASS	3.002
setBacklash	PASS	3.017	PASS	3.011
setInvalidHighBacklashFails	PASS	3.011	PASS	3.001
setInvalidLowBacklashFails	PASS	3.021	PASS	3.022
<i>MotorFindLimitTest</i>				
findLimitWhenMotorStoppedFails	PASS	7.269	PASS	15.180
findLimitWhileBusyFails	PASS	6.587	PASS	8.104
findLimitWithInvalidKeyFails	PASS	3.067	PASS	3.062
findLimitWithValidKeySucceeds	PASS	28.230	PASS	30.734
findNegativeLimitSucceeds	PASS	34.215	PASS	36.725
findNegativeLimitWhenAtNegativeLimitSucceeds	PASS	15.614	PASS	20.240
findPositiveLimitSucceeds	PASS	20.602	PASS	22.357
findPositiveLimitWhenAtPositiveLimitSucceeds	PASS	15.619	PASS	20.202

Sample test results for early run of Java motor

Manual testing, especially in the form of exploratory testing, is still a necessity for testing components

End-to-end automated shots in an emulated NIF environment

Used for frequent verification of shot cycle integrity

- Developed the Automated Shot Tester (AST) engine to run series of end-to-end NIF shot sequences
- Allows coverage of the significantly expanded value of experimental test variations
 - Manual testing could no longer effectively cover all of the variations
- Test cases includes NIF reconfiguration activities, experiment updates mid-shot, and off-normal events
- Test cases are run after-hours
- Focused manual shot testing by developers and testers offline is still required to look deep into subsystem behavior

The screenshot shows the ICCS Shot Inspector interface. At the top, there's a navigation bar with links for Home, Dashboards, Search, Reports, and Help. Below the navigation is a search bar and a toolbar with various icons.

Shots Found (2017-06-15 00:00:00 to 2017-09-13 13:47:44)

experimentID	shotidPattern	shotid	time	ShotDirector	shotid	Action	Activity
1730_Mixed_Fates_ARC_S02a	N170801-007*	N170801-007-0000	2017-07-26 11:42:22	NIF_SHOT	N170726-001-000	Experiment	1730_Mixed_Fates_S01a.SYSTEM_SHOT
1730_Mixed_Fates_ARC_S03a	N170801-008*	N170801-008-0000	2017-07-26 12:51:03	NIF_SHOT	N170726-001-000	GoToState	SS_READY_CYCLE
1730_Mixed_Fates_ARC_S03a	N170801-009*	N170801-009-0001	2017-07-26 12:53:44	NIF_SHOT	N170726-001-000	GoToState	SS_IMPLEMENT_PLAN
1730_Mixed_Fates_ARC_S03a	N170801-005*	N170801-005-0000	2017-07-26 12:53:51	NIF_SHOT	N170726-001-000	GoToState	SS_IMPLEMENT_PLAN
1730_Mixed_Fates_S01a	N170801-004*	N170801-004-0000	2017-07-26 12:58:13	NIF_SHOT	N170726-001-000	ShotType	Rod
1730_Mixed_Fates_S01a	N170801-004*	N170801-004-0001	2017-07-26 12:58:45	NIF_SHOT	N170726-001-000	ShotType	Rod
1730_Sequential_Rod_S01a	N170731-003*	N170731-003-0000	2017-07-26 12:59:00	NIF_SHOT	N170726-001-000	GoToState	SS_PRE_COUNTDOWN
1730_Sequential_Rod_S01a	N170731-002*	N170731-002-0000	2017-07-26 12:59:11	NIF_SHOT	N170726-001-000	Abandon	SS_PRE_COUNTDOWN
1730_Sequential_Rod_S01a	N170731-001*	N170731-001-0000	2017-07-26 13:48:51	NIF_SHOT	N170726-001-000	GoToState	SS_END_SHOT_CYCLE
1730_Mixed_Fates_S01a	N170726-001*	N170726-001-0000	2017-07-26 13:48:51	NIF_SHOT	N170726-001-000	GoToState	SS_END_SHOT_CYCLE
1730_Mixed_Fates_S01a	N170725-001*	N170725-001-0000	2017-07-26 13:48:51	NIF_SHOT	N170726-001-000	GoToState	SS_END_SHOT_CYCLE
1730_Mixed_Fates_S03a	N170720-004*	N170720-004-0000	2017-07-26 13:48:51	NIF_SHOT	N170726-001-000	GoToState	SS_END_SHOT_CYCLE

Macro Step Errors - select a macro step to display system E_LOGs up to the time of the event.

Macro Steps with Errors (N170726-001*)

time	subsystem	location	macrostep	details
2017-07-26 12:53:58.779	TDS	TA	Settings	Phase [PERFORM_SEQUENCE] Step [Head Shot Goals and Settings] Failed to complete successfully user aborts APPLICABLE
2017-07-26 12:53:58.781	TDS	TA	Settings	Phase [PERFORM_SEQUENCE] Step [Head Shot Goals and Settings] Failed to complete successfully user aborts APPLICABLE
2017-07-26 12:53:58.781	PCS	Bu21	PCS_Leak_Test	Phase [PERFORM_SEQUENCE] Step [Leak Test] Failed to

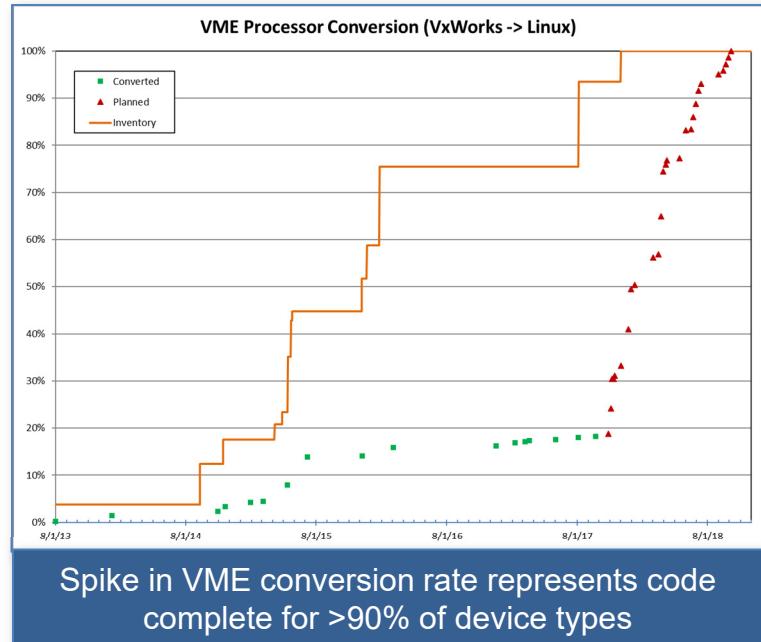
Shot Alerts [Significant] (N170726-001*)

Time	Priority	Minor ID	Alert
2017-07-26 11:41:52	1	SHOT_SETUP_TDTIMING_CALCS_FAILED	[java.lang.Exception: WARNING unable to set timing for DISC T0390-348 Proper part not installed or LoCoS dataset may be missing or incomplete. Details: [Warning: DISC T0390-348: No equipment delay calibration found for TA supervisor. Supervisor Graph: T0390-348 with sweep to 10.]]
2017-07-26 12:53:26	1	SHOT_SETUP_TDTIMING_CALCS_FAILED	[java.lang.Exception: TIMING_CALC_WARNING: Warning unable to set timing for DISC T0390-348 Proper part not installed or LoCoS dataset may be missing or incomplete. Details: [Warning: DISC T0390-348: No equipment delay calibration found for TA supervisor. Supervisor Graph: T0390-348 with sweep to 10.]]
2017-07-26 12:58:11	2	Exception Hold	[select memory Macro Step for the TAS905 TA Shot Supervisor on the TA Collaboration Supervisor Graph. Hold will clear on re-entry selection [pausing]]

Future enhancements includes measurement of performance impact for critical path analysis, and inclusion of AST in our continuous integration environment

Simplification of our build environment through Java porting

- Porting effort has been challenging, but necessary for maintaining ICCS in the long term
- Interpreted language reduces compilation for specific operating systems
- Virtualized Linux and Windows for supervisory systems, and some FEPs
- VME based Single Board Computers (SBC) running customizable Linux Gentoo distribution

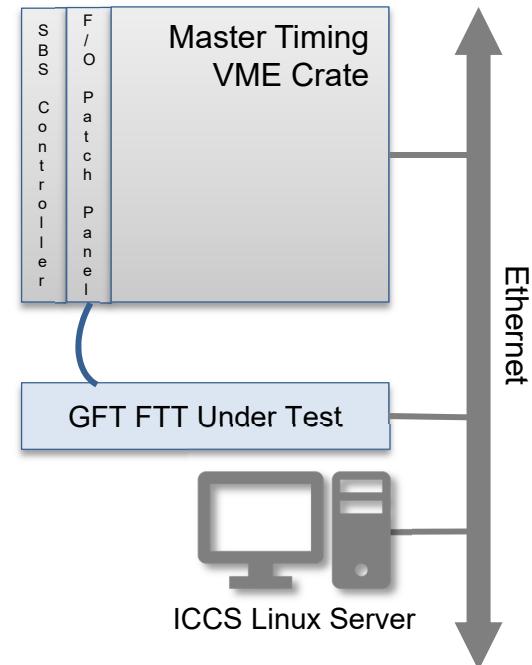


SBC procurements performed in bulk to minimize manufacturer changes.

Replacement of original timing hardware from 1990s

Delivered ICCS test platform for vendor qualification on new Facility Timing Transmitter

- Custom Facility Timing Transmitter hardware was long past end-of-life
- Manufacturer had been bought and resold multiple times. No engagement from new company without \$\$\$
- High risk in source code held in escrow. Many updates had been applied to production FTT firmware
- Loss of timing SME due to retirement
- Greenfield Technology contracted to build drop-in FTT replacement
- Shipped “ICCS In-a-box” complete with automated test suite to factory acceptance testing of new system



Automated test suite provided the interface control requirements in the absence of formal documentation

Greenfield Technologies FTT was successfully deployed and commissioned on NIF following offline qualification



Conclusions

- Modernization is a necessity for sustaining ICCS over the next 20 years
- Cannot rely on retention of SMEs
- Automated testing used to document requirements and behavior of components
 - Supports component changes and onboarding of new employees
- Automated testing required for systems undergoing significant architectural changes
- Flexibility in platforms while simplifying runtime and compile time environments