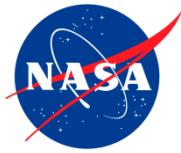


Configuration Testing of Flight Software

Charles Song
Dharmalingam Ganesan
Mikael Lindvall

David McComas
Nicholas Yanchik
Alan Cudmore
Steve Slegel

Fraunhofer USA
Center for Experimental Software Engineering

Goddard Space Flight Center
Code 582, Flight Software Systems Branch

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Motivation

- Flight software is configurable
  - Users select values for **configuration options** for the specific missions and platforms
  - Allows code to be reusable, portable, extensible
- Flight software is also mission-critical
  - Needs to be extensively tested and bug free

- Problem: exponential explosion of **configuration space**
  - Tremendously large number of valid configurations

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

System Under Test

- **Core Flight Software System (CFS):** a flight software (FSW) environment integrating a reusable core flight executive (cFE)
  - A software product-line developed by the NASA Goddard Space Flight Center (GSFC)
  - Designed to be mission-independent and platform-independent
  - Various missions build flight software on top of CFS
    - E.g. GPM
- CFS is **highly configurable** with hundreds of options
  - E.g. cfe_platform_cfg.h file
    - Options: mem_pool_size, max_pipe_depth, max_event_filters, etc.
  - Easily **billions of configurations** in the configuration space

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Configuration Space

- Every **configuration** should be tested to assure software quality
  - Bugs can be caused by any combinations of option settings, e.g.
    - MAX_PIPE_DEPTH=135 &&
      MAX_EVENT_FILTERS=1200
    - MAX_PIPE_DEPTH=55 &&
      MAX_EVENT_FILTERS=2500 &&
      MEM_POOP_SIZE=500
    - …
- How to ensure every possible combination is bug free?

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

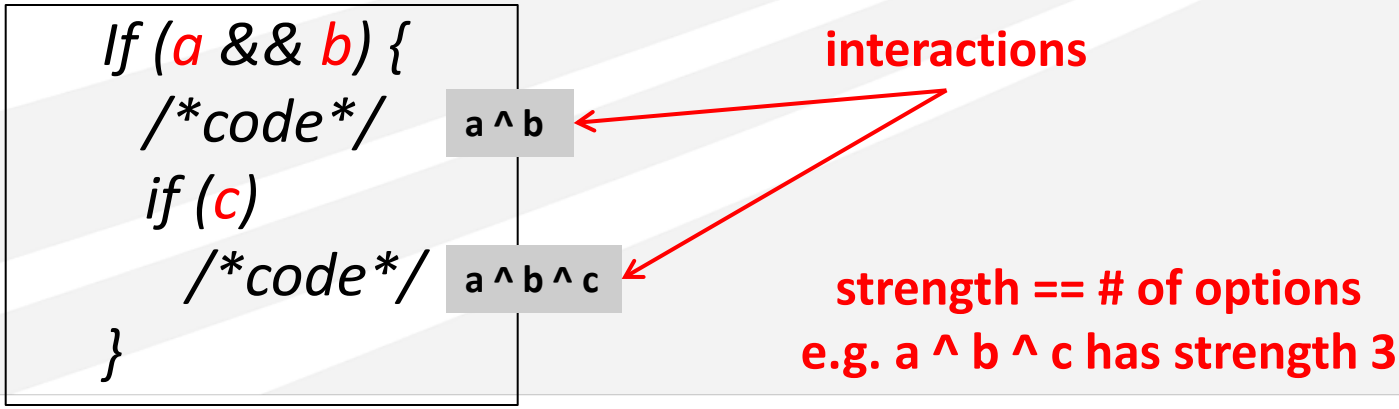
Effective Configuration Space

- Many configurations are essentially executing the same system behaviors
  - Code coverage
  - Test case failures
  - System crashes/hangs
- Goal: only test configurations with different behaviors
  - Minimal set of configurations needed to execute all system behaviors
- Practical goal: efficiently approximate the effective configuration space

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Configuration Interactions

- **Interactions** can tell us about effective configuration space
  - Certain combinations of the options that reveal different system behaviors
  - E.g. complete line coverage



© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Observations on Interactions

- Conducted extensive studies on configuration interactions
  - Interactions are rare and have low strength
  - Most behaviors occur at low strength
  - Higher strength interactions tend to include lower strength ones

Reference:

•Elnatan Reisner, Charles Song, Kin-Keung Ma, Jeffrey S.Foster, Adam Porter, "**Using Symbolic Evaluation to Understand Behavior in Configurable Software Systems**," 32nd International Conference on Software Engineering, May 2010

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

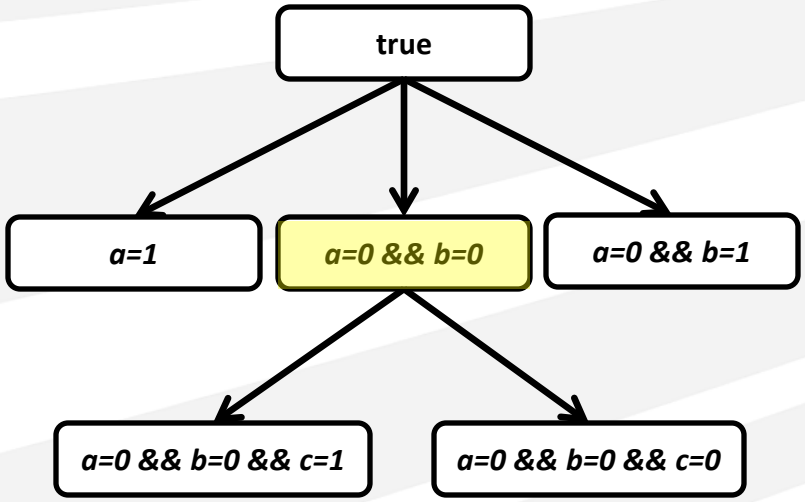
Interaction Tree Discovery (iTree)

- Iteratively search for the **effective configurations** during testing
  - Generates a small **configuration sample** to observe system behavior
  - Uses **machine learning** to discover potential configuration interactions
  - Narrows the **configuration space** of future samples

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Interaction Tree

- **Internal model** of a system's interactions
  - Hierarchical structure of **interactions**



a	b	c	d
0	**0**	0	0
0	**0**	1	1
0	**0**	0	1
0	**0**	1	0
1	1	1	0

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Previous iTree Results

- Empirically evaluated on several industrial systems
  - vsftpd: secure FTP server shipped w/ RedHat
  - ngIRCd: next generation IRC server
  - MySQL: popular open source database, ~1M LoC

Reference:

•Charles Song, Adam Porter, Jeffrey S.Foster, "**iTree: Efficiently Discovering High-Coverage Configurations Using Interaction Trees**," 34th International Conference on Software Engineering, June 2012

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Effectiveness of iTree

- iTree outperformed existing approaches
  - Achieved higher coverage using 3-4 times fewer configurations
  - Executed more system behaviors sooner in the testing process
  - Automatically determined number of configurations to test
  - Programming language independent

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Intelligence of iTree

- iTree predicts combinations of option settings responsible for the different behaviors
  - E.g. Combination of option settings that caused the system to crash
- No other existing approach can produce such information practically
  - ~95% accurate information at almost no cost during configuration testing process
  - Verified results via static analysis

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Scalability of iTree

- iTree's approximation of the **effective configuration space** can still require a lot of testing efforts
  - Analysis/execution of hundreds of configuration
  - Running test cases for each configuration
- Problem: test execution takes time
  - Need to parallelize configuration testing to be practical

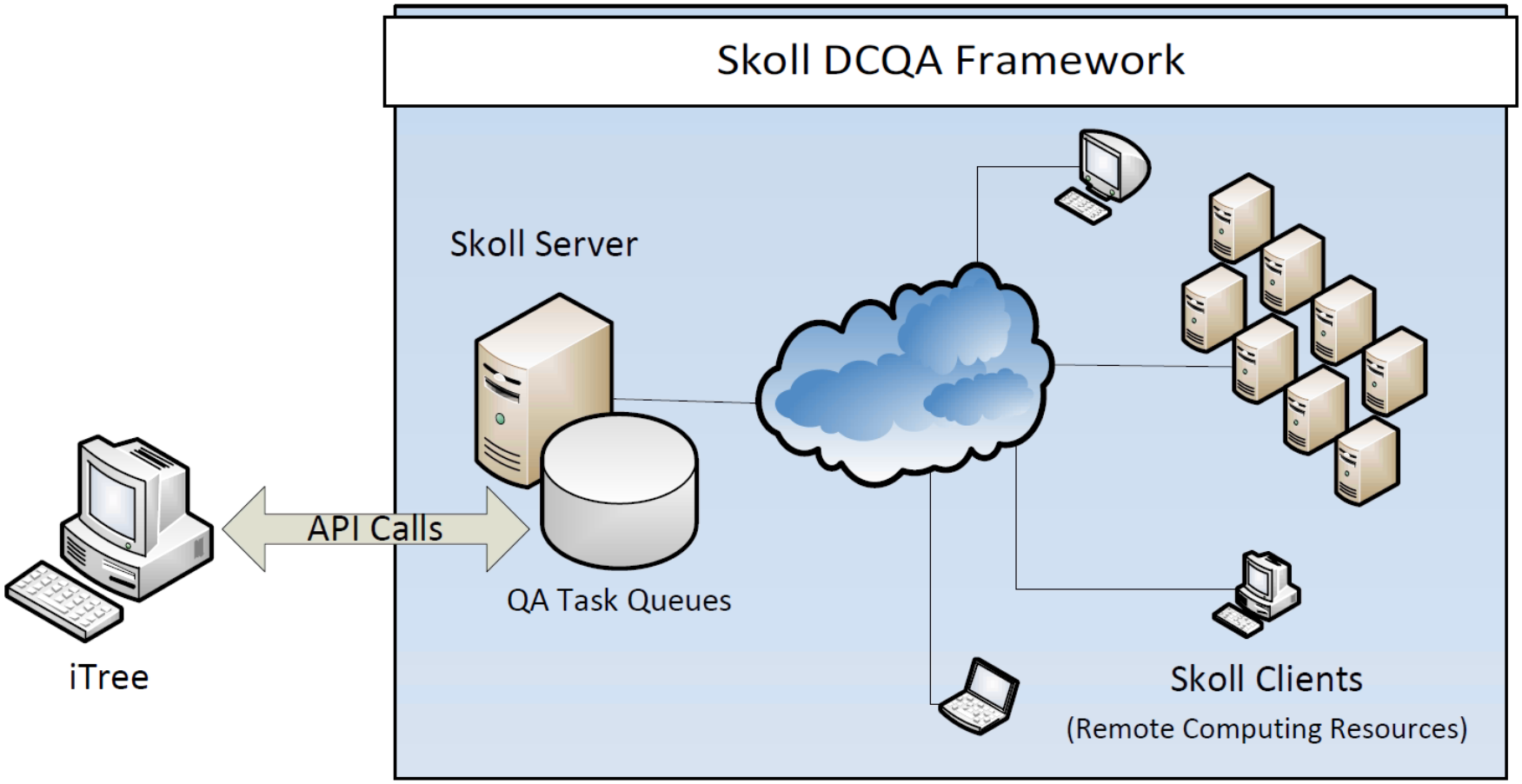
© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Automated Distributed Framework

- Integrated iTree with **Skoll DCQA** (distributed continuous quality assurance)
  - Assigns configurations to Skoll for testing
  - Retrieves execution results for analysis
- Fast, robust, and available
  - Designed to support configuration testing
  - Used for numerous research projects at UMD
  - Parallelized iTree configuration testing across 100+ cluster machines

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Automated Distributed Framework



© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Testing cFE with iTree

- Executed cFE using **stubs** and **unit tests**
- Configuration space model with 40 options
  - Used **min**, **default** and **max** values
- Created automatic configuration file generator
  - E.g. creates valid cfe_platform_cfg.h file
- Parallelized configuration testing on distributed testing framework

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

cFE Unit Test Findings

- Found several critical bugs during preliminary configuration testing

Module	Bug	Configuration Option	Option Settings
SB_UT	Core Dumps	CFE_SB_HIGHEST_VALID_MSGID	0x00FF
SB_UT	Hangs	CFE_SB_HIGHEST_VALID_MSGID	0xFFFF
SB_UT	Core Dumps	CFE_SB_MAX_PIPE_DEPTH	<10
EVS_UT	Hangs	CFE_EVS_LOG_ON	Not #defined
EVS_UT	Hangs	CFE_EVS_DEFAULT_TYPE_FLAG	<=0xD
EVS_UT	Core Dumps	CFE_EVS_MAX_EVENT_FILTERS	>16000
ES_UT	Core Dumps	CFE_ES_MEMPOOL_ALIGNED	Not #defined
ES_UT	Core Dumps	CFE_ES_CDS_MAX_NUM_ENTRIES	>16000

© 2011 Fraunhofer USA, Inc.

Future Work

- Extending configuration testing on cFE
  - Replace cFE's stubs with actual library calls
  - Increase number configuration options and settings
- Extend iTree to support other program behaviors
  - Passing/failing test cases
  - Performance metrics
- Apply iTree to more NASA flight software!
  - Contact me if you are interested

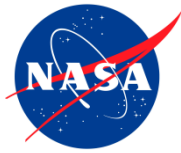
© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Acknowledgement

- Lisa Montgomery - NASA SARP for supporting this applied research
- Sally Godfrey – NASA GSFC

- Thank you!

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering



Questions?

Charles Song (csong@fc-md.umd.edu)

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Acronyms

- CDQA: Distributed Continuous Quality Assurance
- cFE: Core Flight Executive
- FSW: Flight Software
- iTree: Interaction Tree
- SUT: System Under Test
- UMD: University of Maryland

© 2011 Fraunhofer USA, Inc.
Center for Experimental Software Engineering