Performing a Security Review on Unknown Code

A presentation by Imagix and GrammaTech
John Blattner – Imagix, Walter Capitani – GrammaTech | May 2021

Introduction

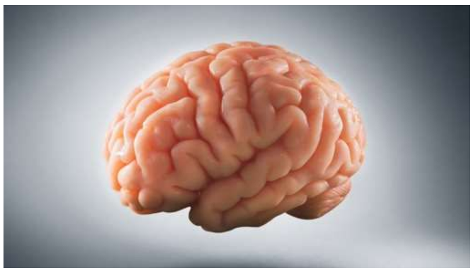
Security Review Challenges

- Large body of source code
- Little to no architecture information
  - Unclear what is where, what is important
- Or you get a delta
  - “has our posture improved”
- You have a short time window
- What tools to use, how do they collaborate

Two Main Problems

Tools That Provide Value

- Static Analysis
- Reverse Engineering
- Code Coverage



What Is Static Analysis?

- Validation of coding standards and best practices
  - MISRA, JPL, CERT-C, …

Quality Metric

- Static verification using formal method concepts to find defects
  - Runtime Errors – Buffer overrun, ..
  - API Misuse – Misuse of socket API, …
  - Suspicious behavior – Dead code, unused variables, …

Security Metric

The Power of Abstract Interpretation

Imagix

- Analyzes paths through the source code
- Without the need for test cases
- With the goal of detecting undefined behavior
  - Which is at the root of many security issues

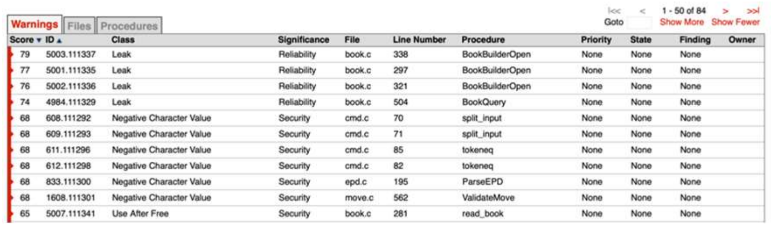
CODESonar®

GRAMMATECH

Static Analysis Is Easy

Imagix

- In: Source Code
- Out: List of warnings
  - Location
  - Class / Category
  - Severity & Complexity
  - Path
  - Relation to the rest of the program



Score ▾	ID ▴	Class	Significance	File	Line Number	Procedure	Priority	State	Finding	Owner
79	5003.111337	Leak	Reliability	book.c	338	BookBuilderOpen	None	None	None	
77	5001.111335	Leak	Reliability	book.c	297	BookBuilderOpen	None	None	None	
76	5002.111336	Leak	Reliability	book.c	321	BookBuilderOpen	None	None	None	
74	4984.111329	Leak	Reliability	book.c	504	BookQuery	None	None	None	
68	608.111292	Negative Character Value	Security	cmd.c	70	split_input	None	None	None	
68	609.111293	Negative Character Value	Security	cmd.c	71	split_input	None	None	None	
68	611.111296	Negative Character Value	Security	cmd.c	85	tokeneq	None	None	None	
68	612.111298	Negative Character Value	Security	cmd.c	82	tokeneq	None	None	None	
68	833.111300	Negative Character Value	Security	epd.c	195	ParseEPD	None	None	None	
68	1608.111301	Negative Character Value	Security	move.c	562	ValidateMove	None	None	None	
65	5007.111341	Use After Free	Security	book.c	281	read_book	None	None	None	

GRAMMATECH

Location / Class / Severity

Imagix

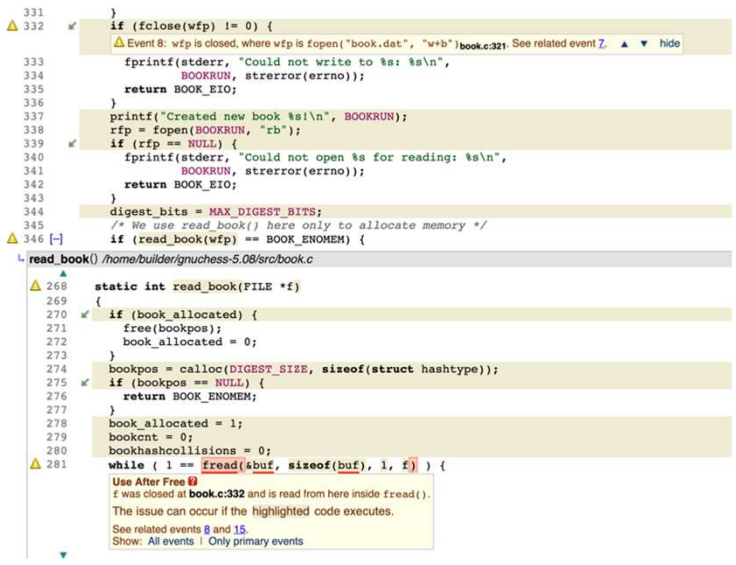
- Class: Buffer Overrun
- Category: Security
- Location: a.c line 67
- Severity / Complexity: 76 (out of 100)

- Can be annotated
- Can be tracked across builds

GRAMMATECH

Path

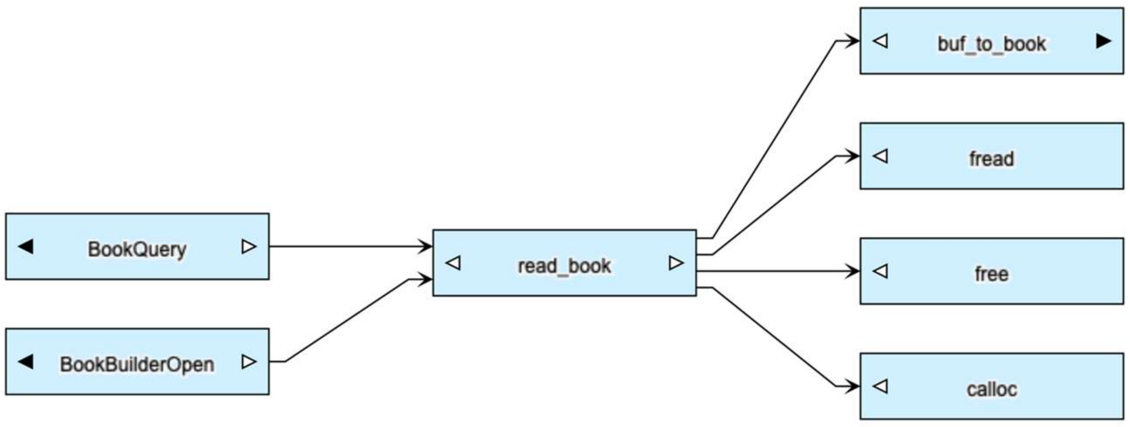
Imagix



```
331        }
332        if (fclose(wfp) != 0) {
           Event 8: wfp is closed, where wfp is fopen("book.dat", "w+b") book.c:321. See related event 7.  ▲ ▼ hide
333          fprintf(stderr, "Could not write to %s: %s\n",
334                  BOOKRUN, strerror(errno));
335          return BOOK_EIO;
336        }
337        printf("Created new book %s!\n", BOOKRUN);
338        rfp = fopen(BOOKRUN, "rb");
339        if (rfp == NULL) {
340          fprintf(stderr, "Could not open %s for reading: %s\n",
341                  BOOKRUN, strerror(errno));
342          return BOOK_EIO;
343        }
344        digest_bits = MAX_DIGEST_BITS;
345        /* We use read_book() here only to allocate memory */
346 [–]    if (read_book(wfp) == BOOK_ENOMEM) {
  read_book() /home/builder/gnuchess-5.08/src/book.c
    268   static int read_book(FILE *f)
    269   {
    270     if (book_allocated) {
    271       free(bookpos);
    272       book_allocated = 0;
    273     }
    274     bookpos = calloc(DIGEST_SIZE, sizeof(struct hashtype));
    275     if (bookpos == NULL) {
    276       return BOOK_ENOMEM;
    277     }
    278     book_allocated = 1;
    279     bookcnt = 0;
    280     bookhashcollisions = 0;
    281     while ( 1 == fread(&buf, sizeof(buf), 1, f) ) {
          Use After Free
          f was closed at book.c:332 and is read from here inside fread().
          The issue can occur if the highlighted code executes.
          See related events 8 and 15.
          Show: All events | Only primary events
```

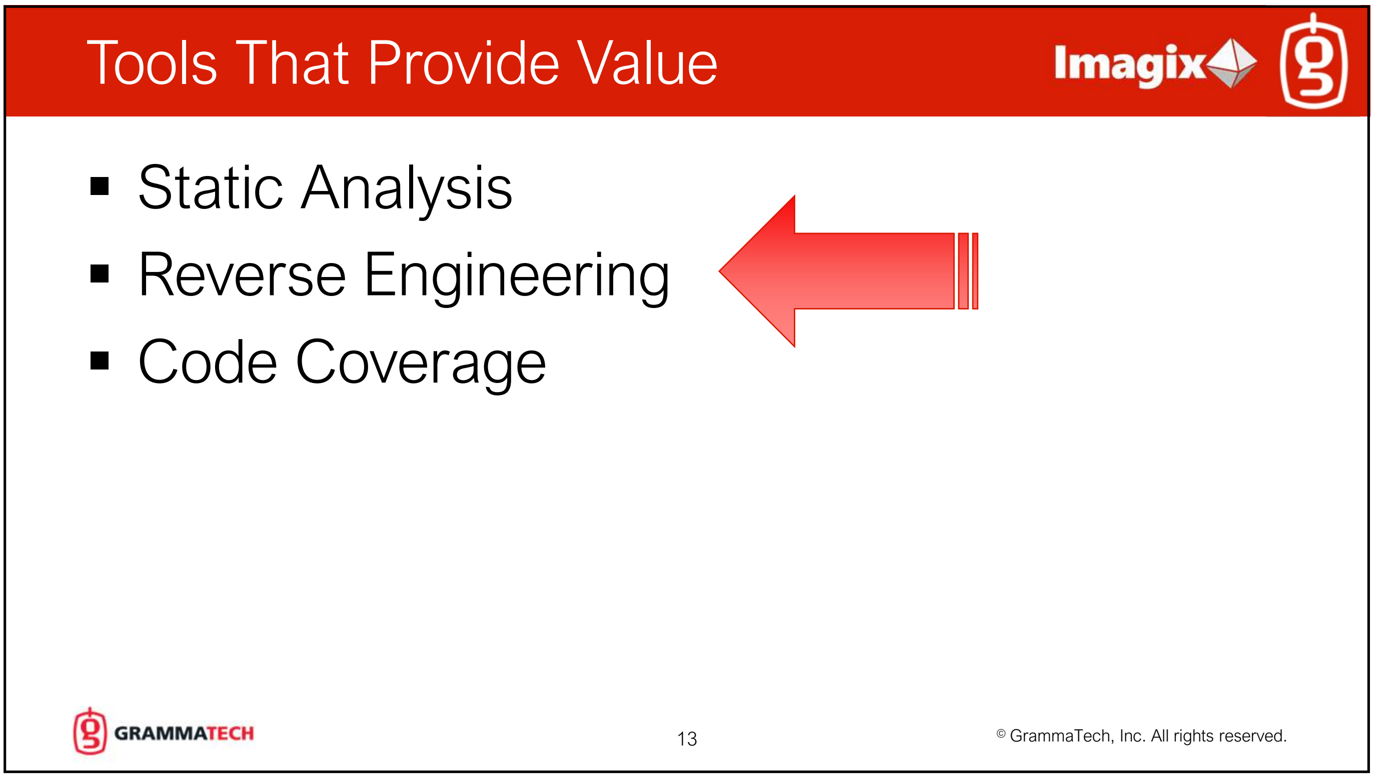
GRAMMATECH

Relation



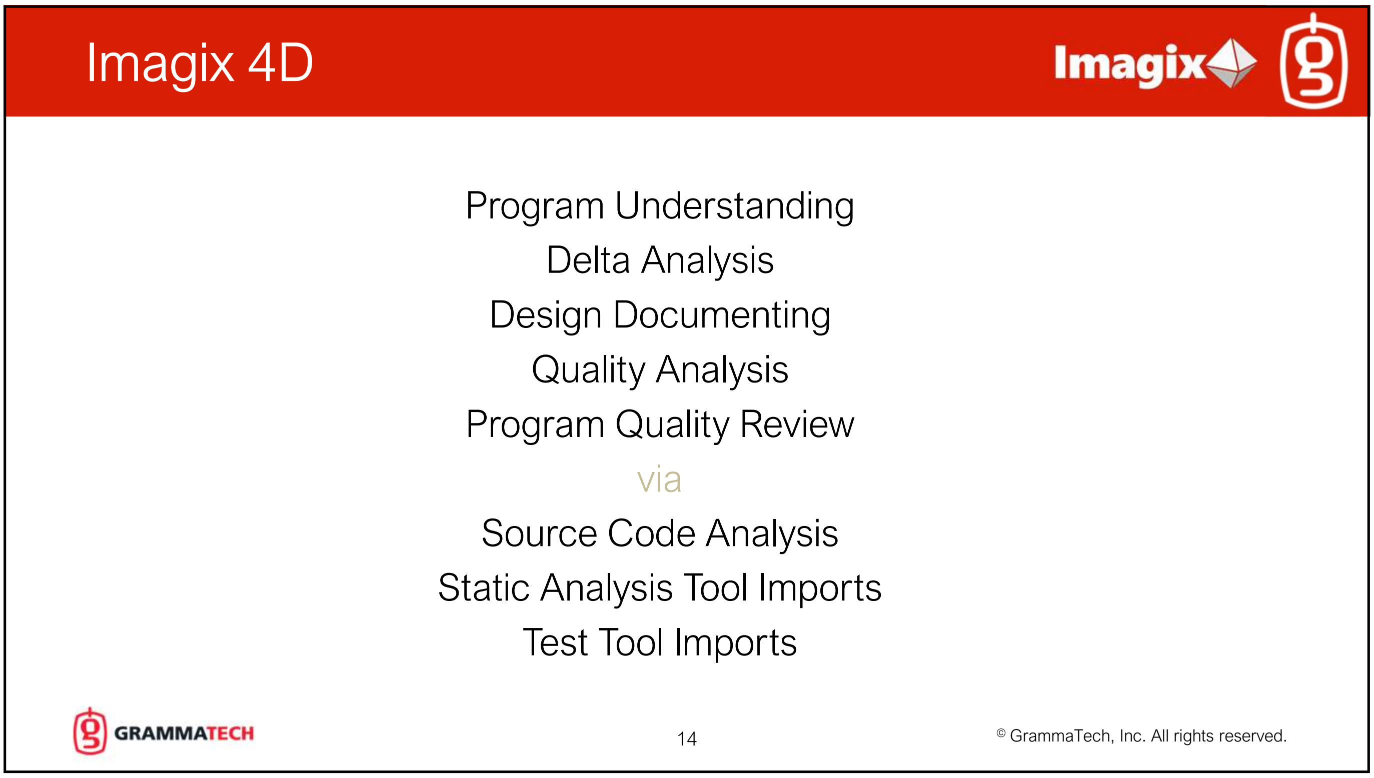
In The Darkness Bind Them

- Static Analysis provides one view
- People use multiple tools
- SARIF is here to collaborate
  – Static Analysis Results Interchange Format
- Export to SARIF and bring in other tools
  – For other views



Tools That Provide Value

- Static Analysis
- Reverse Engineering
- Code Coverage

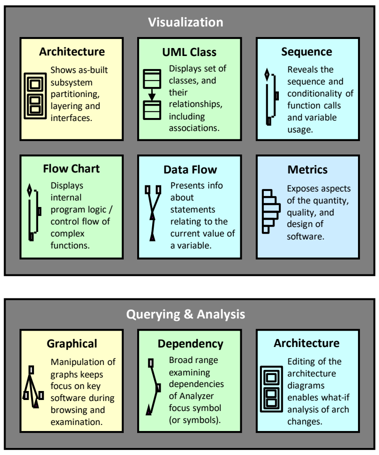


Imagix 4D

Program Understanding
Delta Analysis
Design Documenting
Quality Analysis
Program Quality Review
via
Source Code Analysis
Static Analysis Tool Imports
Test Tool Imports

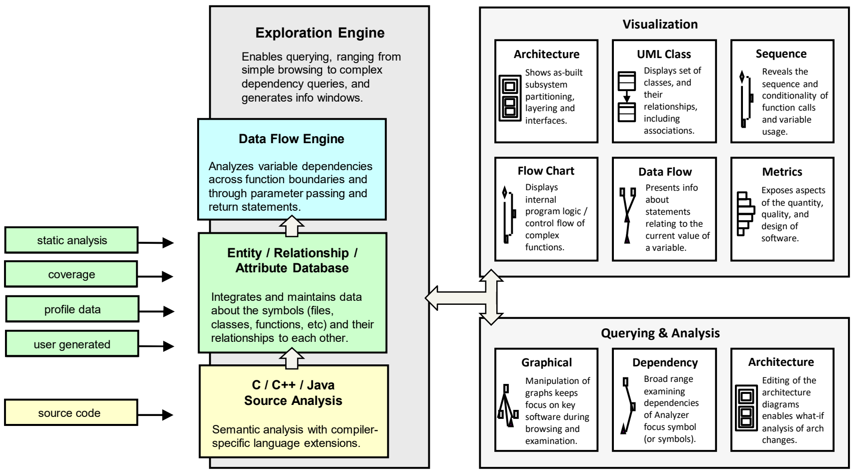
Program Understanding and Analysis

- Provides set of display windows, each optimized for rapid, intuitive understanding of specific type of information
- Uses visualization techniques such as color and data filtering to speed comprehension
- Incorporates querying functionality to focus on areas of interest and tracking across information domains


Visualization
Architecture
Shows as-built subsystem partitioning, layering and interfaces.
UML Class
Displays set of classes, and their relationships, including associations.
Sequence
Reveals the sequence and conditionality of function calls and variable usage.
Flow Chart
Displays internal program logic / control flow of complex functions.
Data Flow
Presents info about statements relating to the current value of a variable.
Metrics
Exposes aspects of the quantity, quality, and design of software.
Querying & Analysis
Graphical
Manipulation of graphs keeps focus on key software during browsing and examination.
Dependency
Broad range examining dependencies of Analyzer focus symbol (or symbols).
Architecture
Editing of the architecture diagrams enables what-if analysis of arch changes.

GRAMMATECH

Program Understanding and Analysis


Exploration Engine
Enables querying, ranging from simple browsing to complex dependency queries, and generates info windows.
Data Flow Engine
Analyzes variable dependencies across function boundaries and through parameter passing and return statements.
Entity / Relationship / Attribute Database
Integrates and maintains data about the symbols (files, classes, functions, etc) and their relationships to each other.
C / C++ / Java Source Analysis
Semantic analysis with compiler-specific language extensions.
static analysis
coverage
profile data
user generated
source code
Visualization
Architecture
Shows as-built subsystem partitioning, layering and interfaces.
UML Class
Displays set of classes, and their relationships, including associations.
Sequence
Reveals the sequence and conditionality of function calls and variable usage.
Flow Chart
Displays internal program logic / control flow of complex functions.
Data Flow
Presents info about statements relating to the current value of a variable.
Metrics
Exposes aspects of the quantity, quality, and design of software.
Querying & Analysis
Graphical
Manipulation of graphs keeps focus on key software during browsing and examination.
Dependency
Broad range examining dependencies of Analyzer focus symbol (or symbols).
Architecture
Editing of the architecture diagrams enables what-if analysis of arch changes.

GRAMMATECH

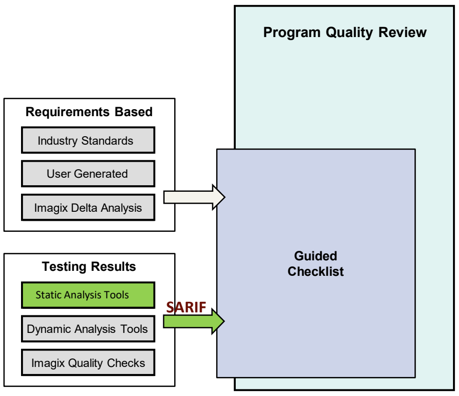
Program Quality Review

- Considerable initiatives exist to improve software reliability using checklists
  - CWE, MISRA, C-Cert, ISO xxx, HIS, PCI-DSS, …
- Some checks can be evaluated by high-end static analyzers
  - Result in lists of potential violations needing review
  - Show violations that are otherwise mitigated or false positives
  - Need careful analysis of potential fixes so nothing else breaks
- Some checks require dynamic testing (e.g. code coverage)
- Third category of checks can only be done interactively

- In all these cases, software checklist compliance is very tedious and incomplete without adequate tool support
  - Reviewing software at source code level is needle in haystack
  - Mining design-level information and code to guide review
  - Recording of steps taken and artifacts to make review repeatable
  - Comprehensive reporting of checklist review status and results

Program Quality Review

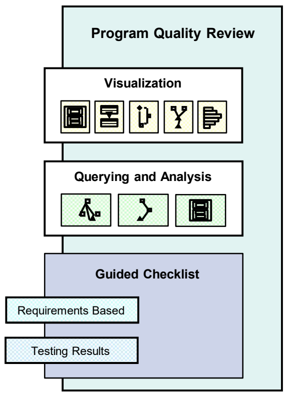
- Creates reviews for software projects from standards and from results of test tools



Program Quality Review

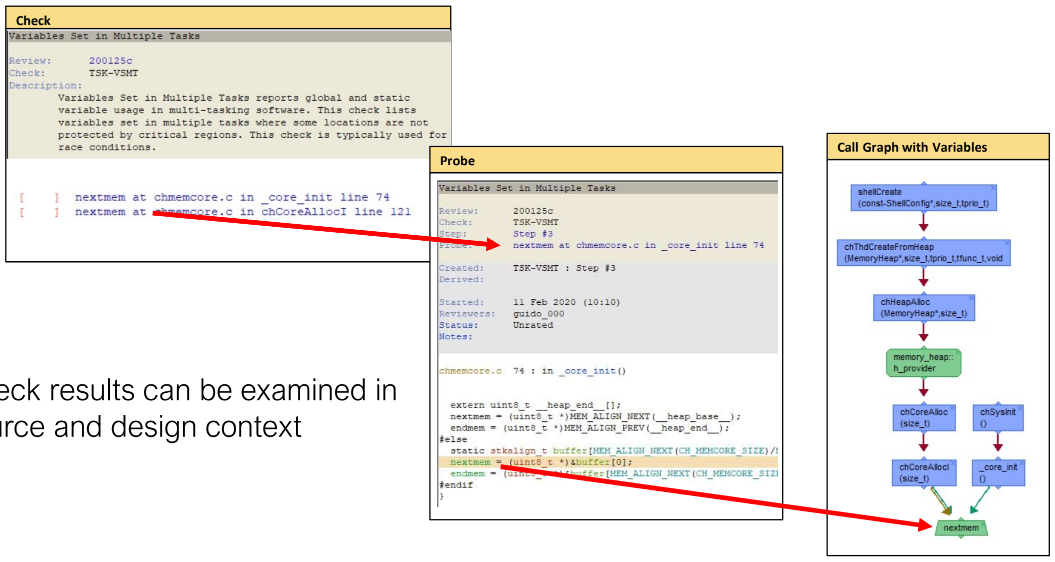
Imagix

- Creates reviews for software projects from standards and from results of test tools
- Uses visualization to enable rapid understanding of check findings
- Represents checks in navigable database with structural design information
- Produces metrics and automated checks to help in checklist review
- Guides reviewers through checklists and records each review step and results
- Reports on checklist review progress and results



Using Exploration Tools with Reviews

Imagix


Check

```
Variables Set in Multiple Tasks

Review:      200125c
Check:       TSK-VSMT
Description:
        Variables Set in Multiple Tasks reports global and static
        variable usage in multi-tasking software. This check lists
        variables set in multiple tasks where some locations are not
        protected by critical regions. This check is typically used for
        race conditions.

[    ]  nextmem at chmemcore.c in _core_init line 74
[    ]  nextmem at chmemcore.c in chCoreAllocI line 121
```

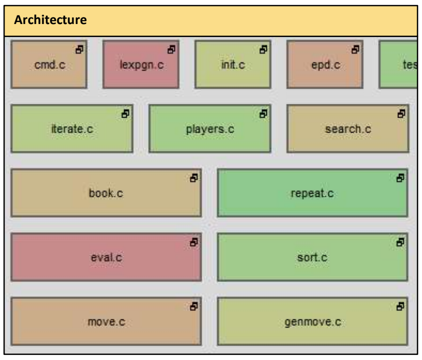
Probe

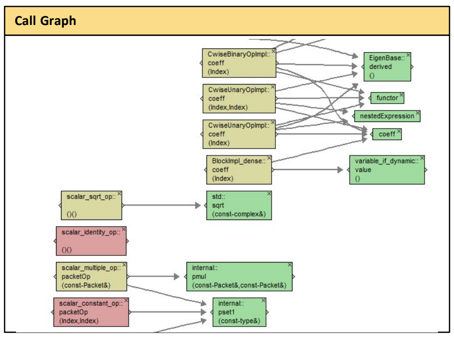
```
Variables Set in Multiple Tasks

Review:      200125c
Check:       TSK-VSMT
Step:        Step #3
Probe:       nextmem at chmemcore.c in _core_init line 74

Created:     TSK-VSMT : Step #3
Derived:

Started:     11 Feb 2020 (10:10)
Reviewers:   guido_000
Status:      Unrated
Notes:

chmemcore.c  74 : in _core_init()


  extern uint8_t __heap_end__[];
  nextmem = (uint8_t *)MEM_ALIGN_NEXT(__heap_base__);
  endmem = (uint8_t *)MEM_ALIGN_PREV(__heap_end__);
#else
  static stkalign_t buffer[MEM_ALIGN_NEXT(CH_MEMCORE_SIZE)/
  nextmem = (uint8_t *)&buffer[0];
  endmem = (uint8_t *)&buffer[MEM_ALIGN_NEXT(CH_MEMCORE_SIZE
#endif
}
```

Call Graph with Variables

- Check results can be examined in source and design context

Using Exploration Tools with Reviews



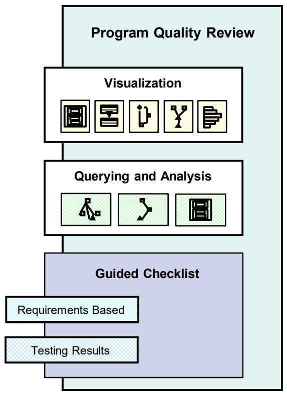

Call Graph
CwiseBinaryOpImpl:: coeff (Index)
CwiseUnaryOpImpl:: coeff (Index,Index)
CwiseUnaryOpImpl:: coeff (Index)
BlockImpl_dense:: coeff (Index)
EigenBase:: derived ()
functor
nestedExpression
coeff
variable_if_dynamic:: value ()
scalar_sqrt_op:: ()()
std:: sqrt (const-complex&)
scalar_identity_op:: ()()
scalar_multiple_op:: packetOp (const-Packet&)
internal:: pmul (const-Packet&,const-Packet&)
scalar_constant_op:: packetOp (Index,Index)
internal:: pset1 (const-type&)

- Check results can be examined in source and design context
- Check results metrics can be applied to function, file, subsystem levels

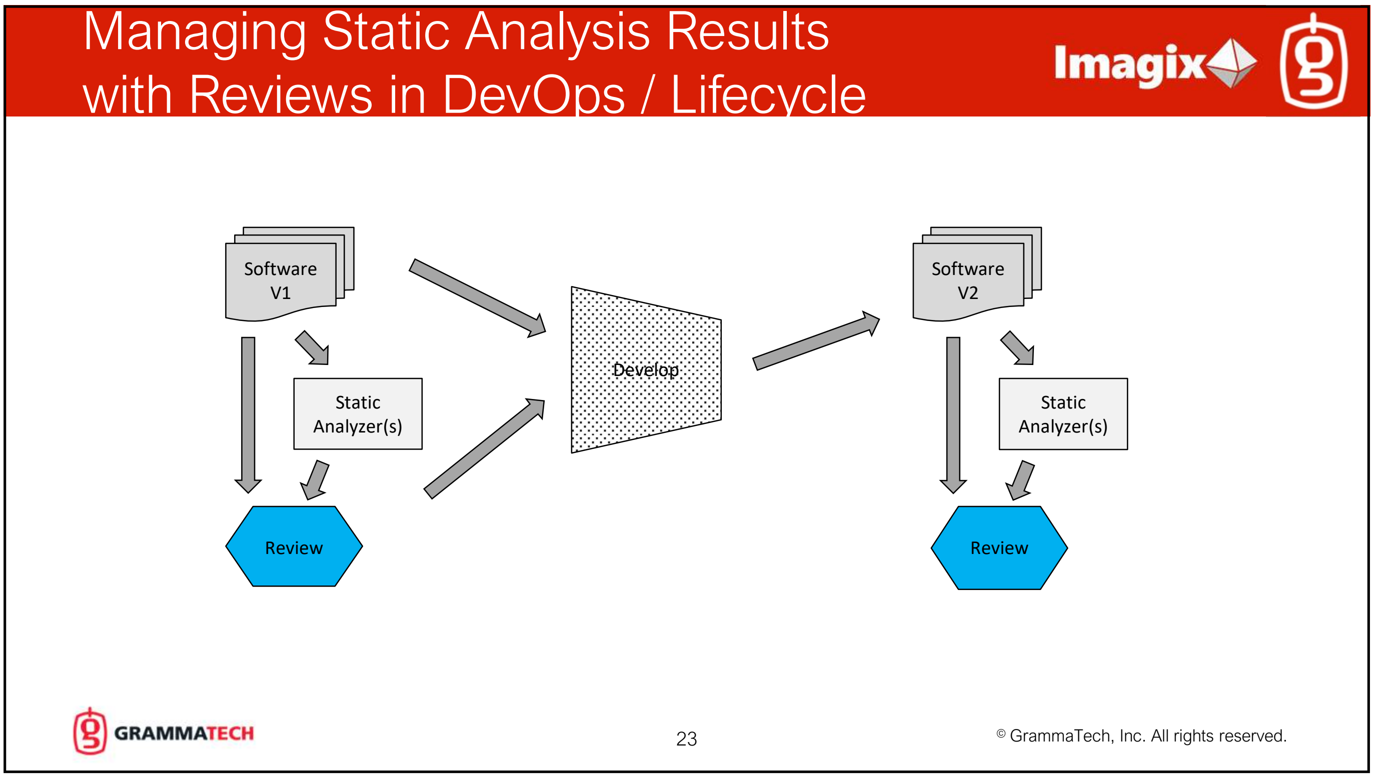
21

Program Quality Review

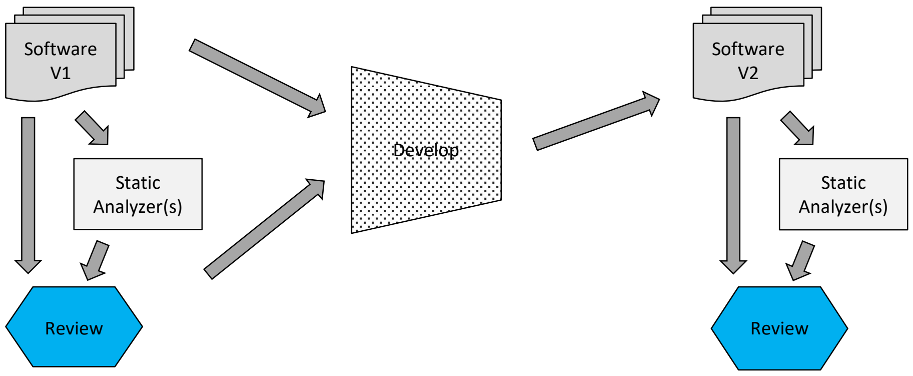
- Creates reviews for software projects from standards and from results of test tools
- Uses visualization to enable rapid understanding of check findings
- Represents checks in navigable database with structural design information
- Produces metrics and automated checks to help in checklist review
- Guides reviewers through checklists and records each review step and results
- Reports on checklist review progress and results
- Reuses reviews for checking versions/variants of code

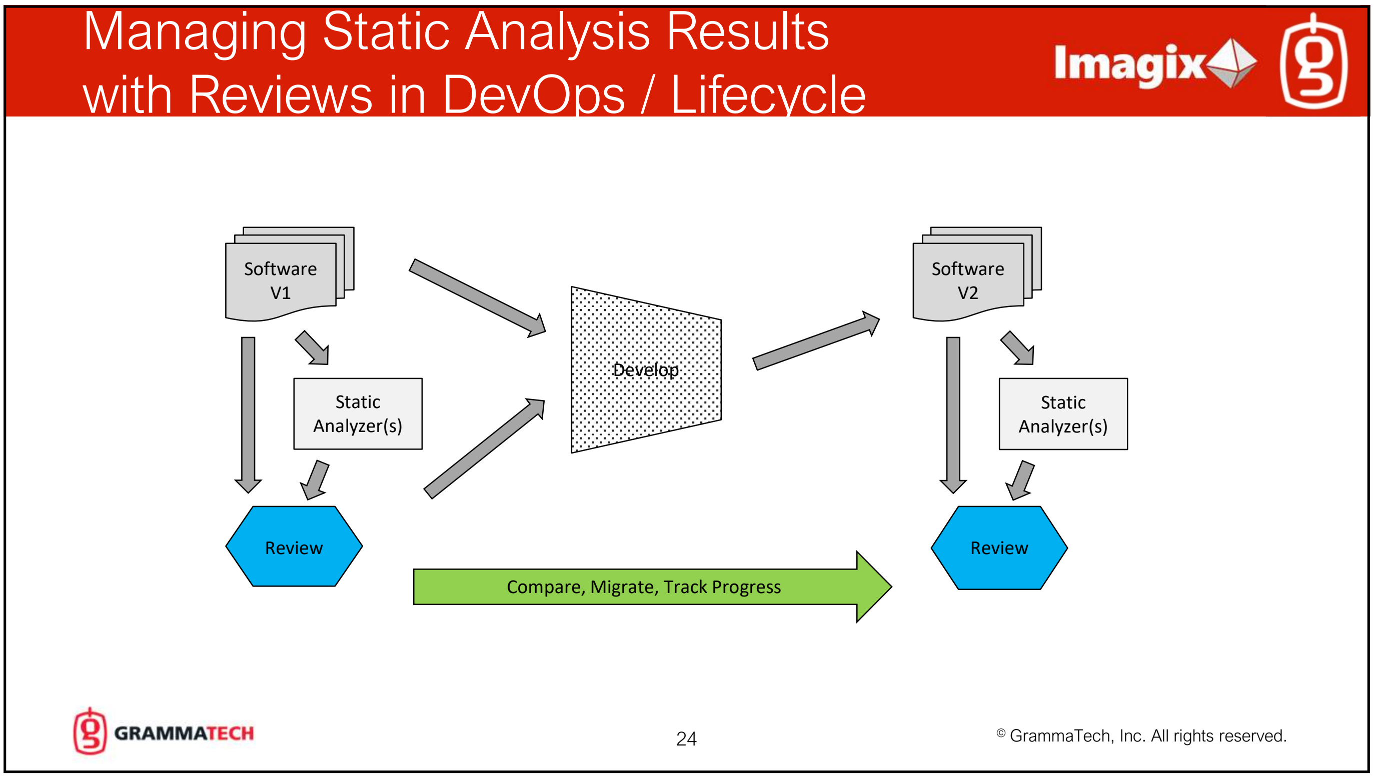


22

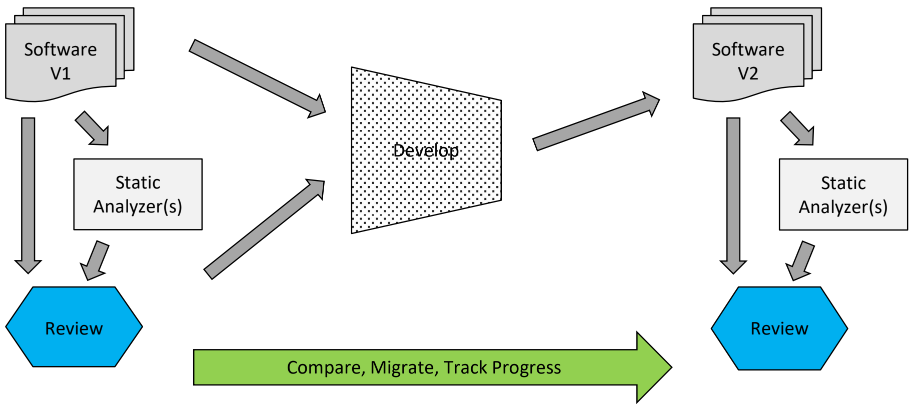


Managing Static Analysis Results with Reviews in DevOps / Lifecycle



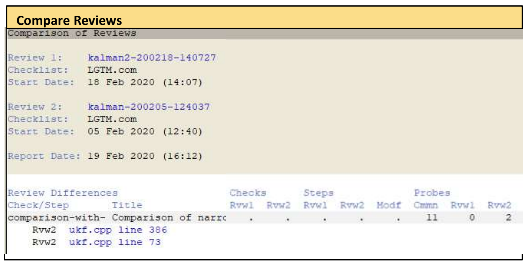


Managing Static Analysis Results with Reviews in DevOps / Lifecycle



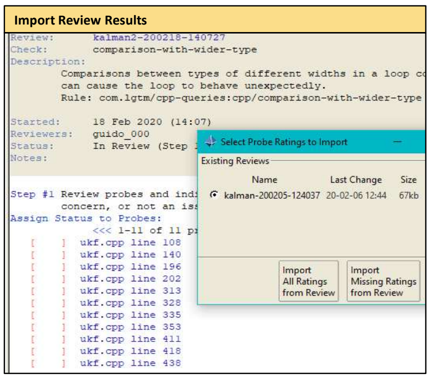
Reviews across Versions

- Deltas of Checks through Reviews
  - Compare review for checks performed on new version with old version
  - Carry over review artifacts and results into new version
  - Structural delta analysis guarantees accurate matching



Compare Reviews

```
Comparison of Reviews

Review 1:    kalman2-200218-140727
Checklist:   LGTM.com
Start Date:  18 Feb 2020 (14:07)

Review 2:    kalman-200205-124037
Checklist:   LGTM.com
Start Date:  05 Feb 2020 (12:40)

Report Date: 19 Feb 2020 (16:12)

Review Differences                        Checks      Steps             Probes
Check/Step       Title                    Rvw1  Rvw2  Rvw1  Rvw2  Modf  Cmmn  Rvw1  Rvw2
comparison-with- Comparison of narrc        .     .     .     .     .    11     0     2
    Rvw2  ukf.cpp line 386
    Rvw2  ukf.cpp line 73
```



Import Review Results

```
Review:       kalman2-200218-140727
Check:        comparison-with-wider-type
Description:
        Comparisons between types of different widths in a loop co
        can cause the loop to behave unexpectedly.
        Rule: com.lgtm/cpp-queries:cpp/comparison-with-wider-type

Started:      18 Feb 2020 (14:07)
Reviewers:    guido_000
Status:       In Review (Step 1
Notes:

Step #1 Review probes and indi
        concern, or not an iss
Assign Status to Probes:
              <<< 1-11 of 11 pr
   [    ]  ukf.cpp line 108
   [    ]  ukf.cpp line 140
   [    ]  ukf.cpp line 196
   [    ]  ukf.cpp line 202
   [    ]  ukf.cpp line 313
   [    ]  ukf.cpp line 328
   [    ]  ukf.cpp line 335
   [    ]  ukf.cpp line 353
   [    ]  ukf.cpp line 411
   [    ]  ukf.cpp line 418
   [    ]  ukf.cpp line 438
```

Select Probe Ratings to Import

Existing Reviews

	Name	Last Change	Size
●	kalman-200205-124037	20-02-06 12:44	67kb

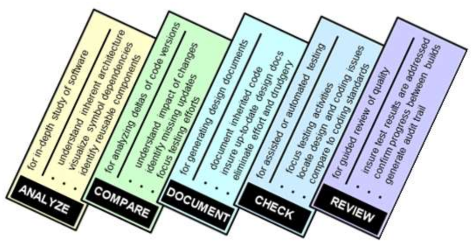
Import All Ratings from Review
Import Missing Ratings from Review

Managing Static Analysis Results with Reviews

- Bring different static analyzer results together
- Manage assessment of potential issues linked into the software
- Ease of communication within the team
- Management reports and audit trail for issues
- Reuse of assessments of issues across versions
- Tracking of progress across versions

Imagix 4D in Summary

- Transforms existing code from a resource drain into a major asset
- Uses visualization to enable rapid, intuitive program understanding
- Provides a broad range of views to enable efficient, effective analysis
- Automatically generates comprehensive design documents
- Produces metrics and automated checks to spot problem areas in design and coding
- Guides systematic review of quality standards and test results


for in-depth study of software
understand inherent architecture
visualize symbol dependencies
identify reusable components
ANALYZE
for analyzing deltas of code versions
understand impact of changes
identify missing updates
focus testing efforts
COMPARE
for generating design documents
document inherited code
insure up-to-date design docs
eliminate effort and drudgery
DOCUMENT
for assisted or automated testing
focus testing activities
locate design and coding issues
compare to coding standards
CHECK
for guided review of quality
insure test results are addressed
confirm progress between builds
generate audit trail
REVIEW

Tools That Provide Value

- Static Analysis
- Reverse Engineering
- Code Coverage

Code Coverage

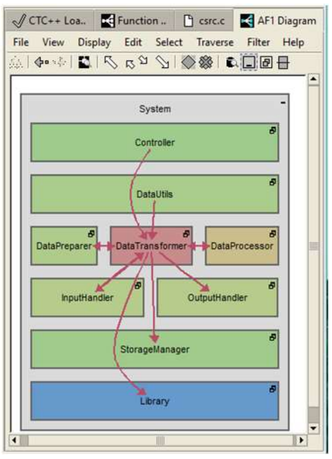
Imagix

- Explains what code was tested
- Overlay this into the architecture model

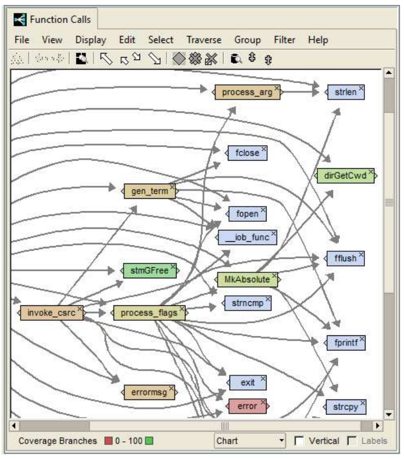
GRAMMATECH

Coverage Visualization

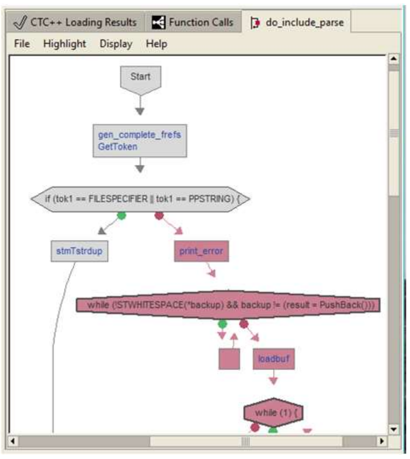
Imagix



Architecture



Control Flow


CTC++ Loading Results
Function Calls
do_include_parse
File Highlight Display Help
Start
gen_complete_frefs
GetToken
if (tok1 == FILESPECIFIER || tok1 == PPSTRING) {
stmTstrdup
print_error
while (!STWHITESPACE(*backup) && backup != (result = PushBack()))
loadbuf
while (1) {

Program Logic

GRAMMATECH

Take Aways

- Reviewing unknown code is hard
- Tools can help
- Static analysis helps find flaws
- Reverse engineering helps visualize
- Code Coverage shows test coverage

Questions and Contact Details

- Happy to answer any questions
  - Live or via email

- John : johnb@imagix.com
- Walter : wcapitani@grammatech.com