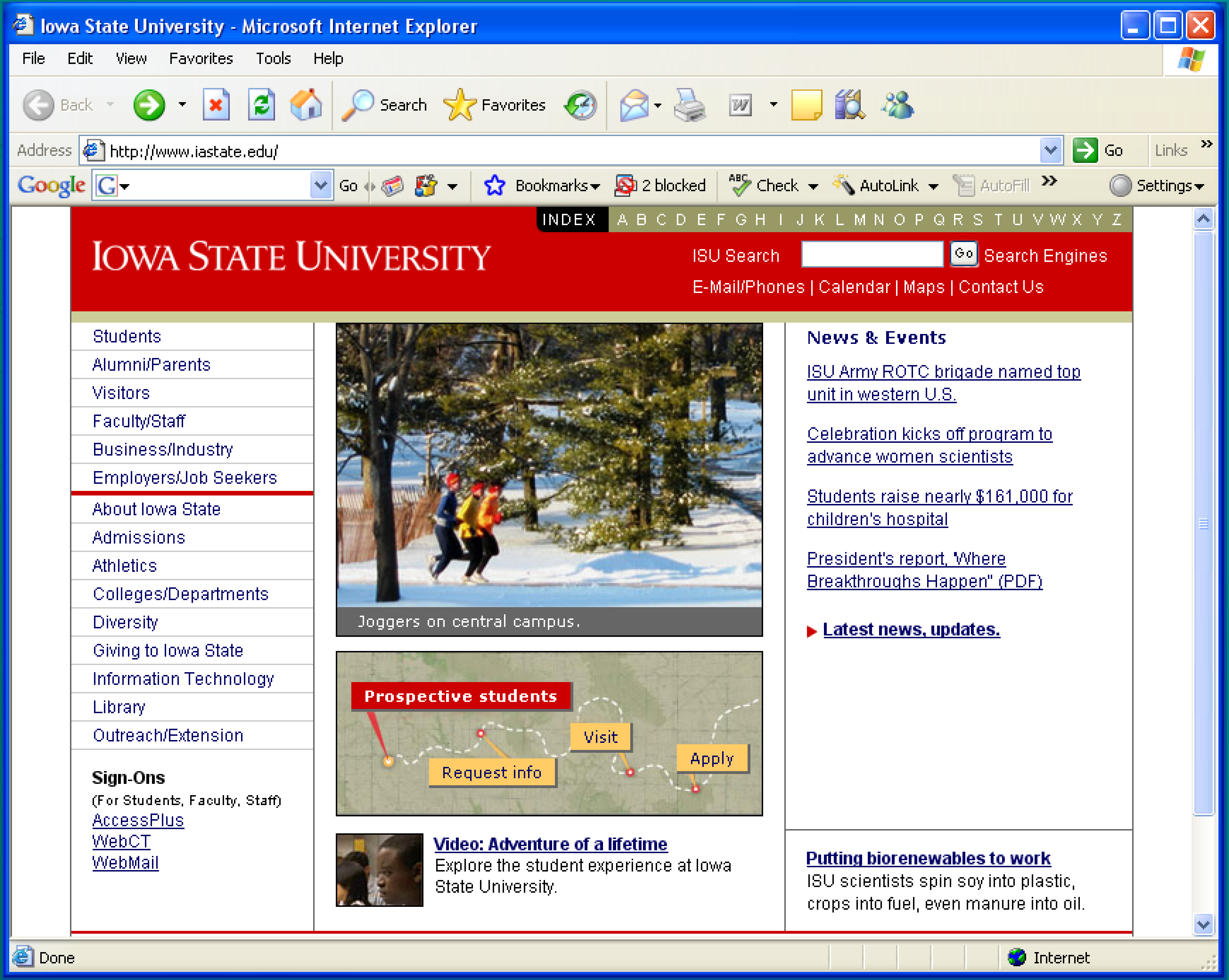
Software Productivity and Reliability Tools and Techniques

Microsoft Presentation, February 5, 2007

S. C. Kothari
Iowa State University & EnSoft Corp.
Phone: 515-294-7212
Email: kothari@iastate.edu



INDEX A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

IOWA STATE UNIVERSITY

ISU Search Go Search Engines

E-Mail/Phones | Calendar | Maps | Contact Us

- Students
- Alumni/Parents
- Visitors
- Faculty/Staff
- Business/Industry
- Employers/Job Seekers
- About Iowa State
- Admissions
- Athletics
- Colleges/Departments
- Diversity
- Giving to Iowa State
- Information Technology
- Library
- Outreach/Extension

Sign-Ons
(For Students, Faculty, Staff)
AccessPlus
WebCT
WebMail

Joggers on central campus.

Prospective students
Request info
Visit
Apply

Video: Adventure of a lifetime
Explore the student experience at Iowa State University.

News & Events

ISU Army ROTC brigade named top unit in western U.S.

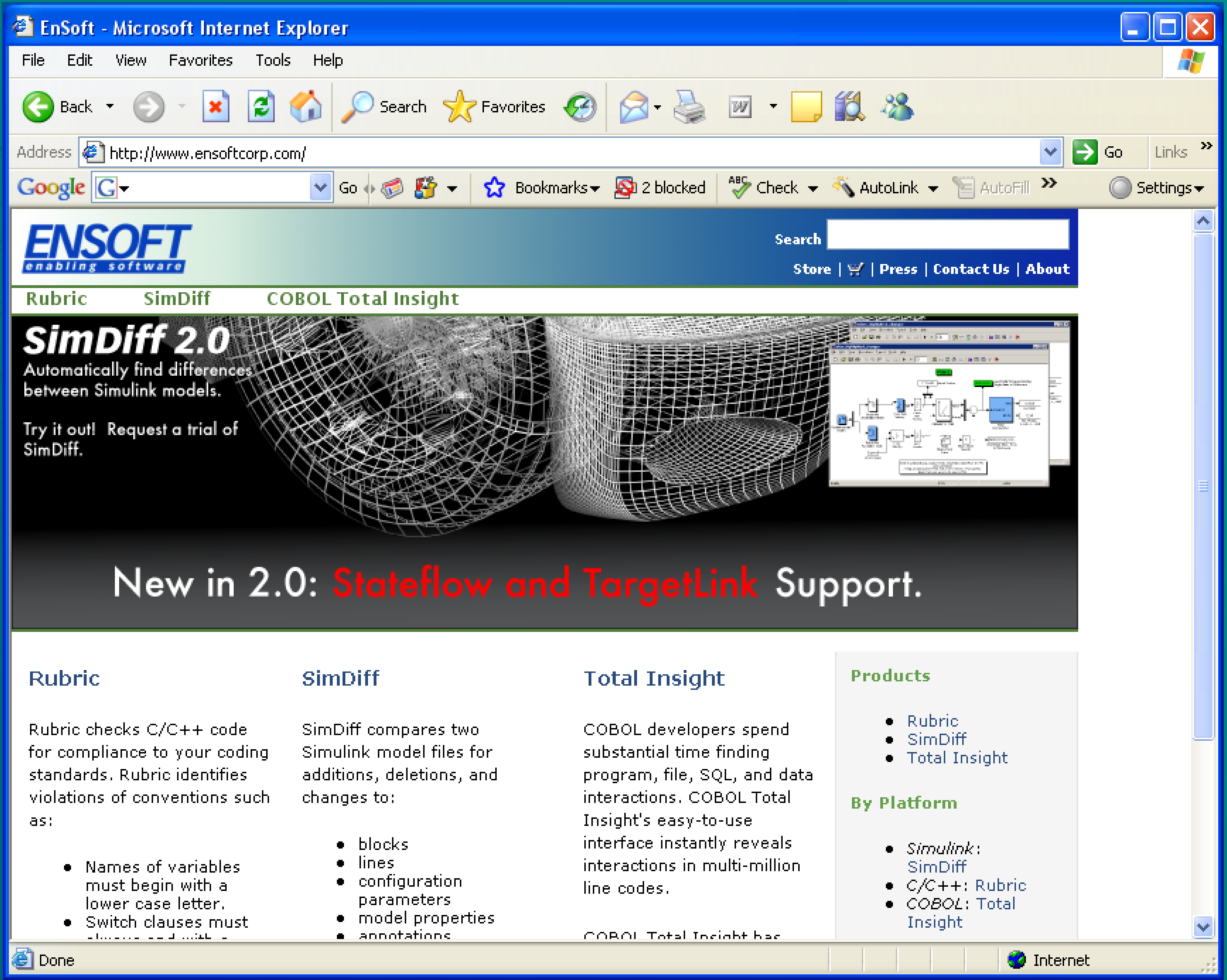
Celebration kicks off program to advance women scientists

Students raise nearly $161,000 for children's hospital

President's report, 'Where Breakthroughs Happen" (PDF)

▶ **Latest news, updates.**

Putting biorenewables to work
ISU scientists spin soy into plastic, crops into fuel, even manure into oil.



SimDiff 2.0

Automatically find differences between Simulink models.

Try it out! Request a trial of SimDiff.

New in 2.0: Stateflow and TargetLink Support.

Rubric

Rubric checks C/C++ code for compliance to your coding standards. Rubric identifies violations of conventions such as:

- Names of variables must begin with a lower case letter.
- Switch clauses must always end with a

SimDiff

SimDiff compares two Simulink model files for additions, deletions, and changes to:

- blocks
- lines
- configuration parameters
- model properties
- annotations

Total Insight

COBOL developers spend substantial time finding program, file, SQL, and data interactions. COBOL Total Insight's easy-to-use interface instantly reveals interactions in multi-million line codes.

COBOL Total Insight has

Products

- Rubric
- SimDiff
- Total Insight

By Platform

- *Simulink*: SimDiff
- *C/C++*: Rubric
- *COBOL*: Total Insight

Bygone Era

...New Reality

The IBM 360 finally reached the Palouse wheat fields of Eastern Washington. It cost the university two million dollars in 1966. I began using it and the world opened up. Right away, I solved a differential equation numerically and found myself staring at a Butterfly Effect. It’d be decades before that term entered our vocabulary. These new machines led us into a brave new world.

– from *Engines of Our Ingenuity* by John H. Lienhard

IBM 360

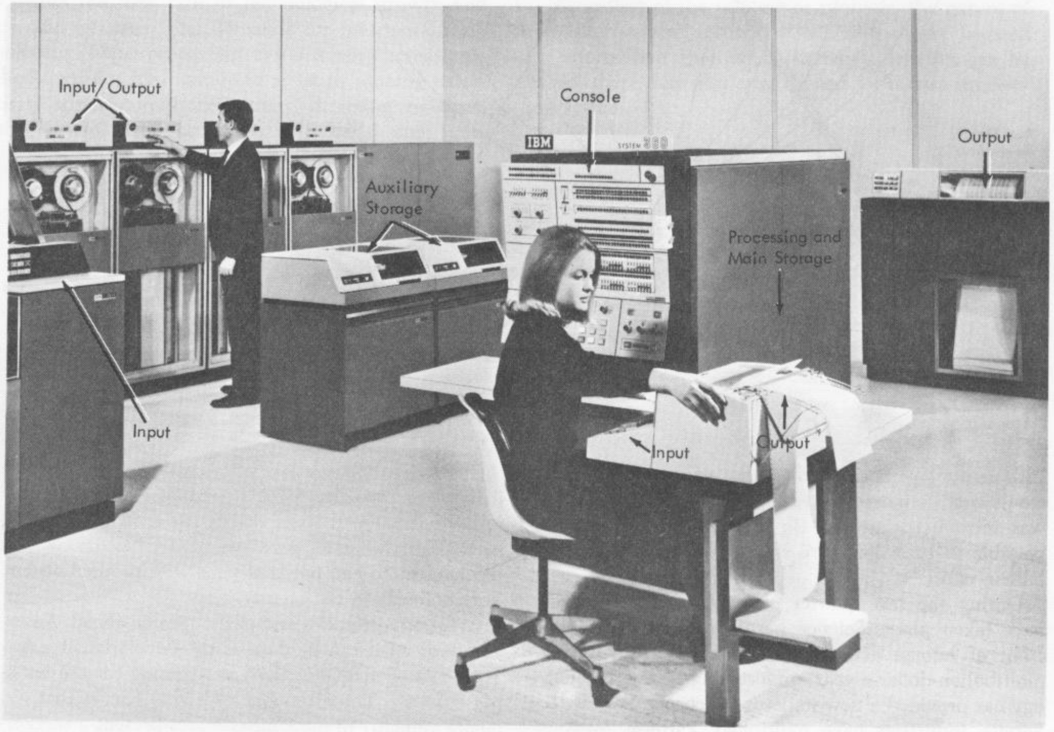


Figure 1. IBM System/360 Model 40 Data Processing System

Operating System/360

- The not-unexpected passing away of OS/360 in its 21st release – August 2, 1972.
- Obituary:

  The offspring first saw the light of day in December 1965 and the birth announcement recorded a weight of 64K. It rapidly became apparent that OS, in spite of its unusual size, was more than normally subject to childhood diseases. For a long period, this weak and sickly baby hovered close to death despite almost continuous transformations and major transplants of several vital organs. Many experts are of the opinion that the huge weight of OS at birth contributed greatly to its early ill health. OS is survived by two lineal descendants, OS/VS1 and OS/VS2. It will be mourned by its many friends and particularly by the over 10,000 system programmers throughout the world who owe their jobs to its existence.

Bygone Era

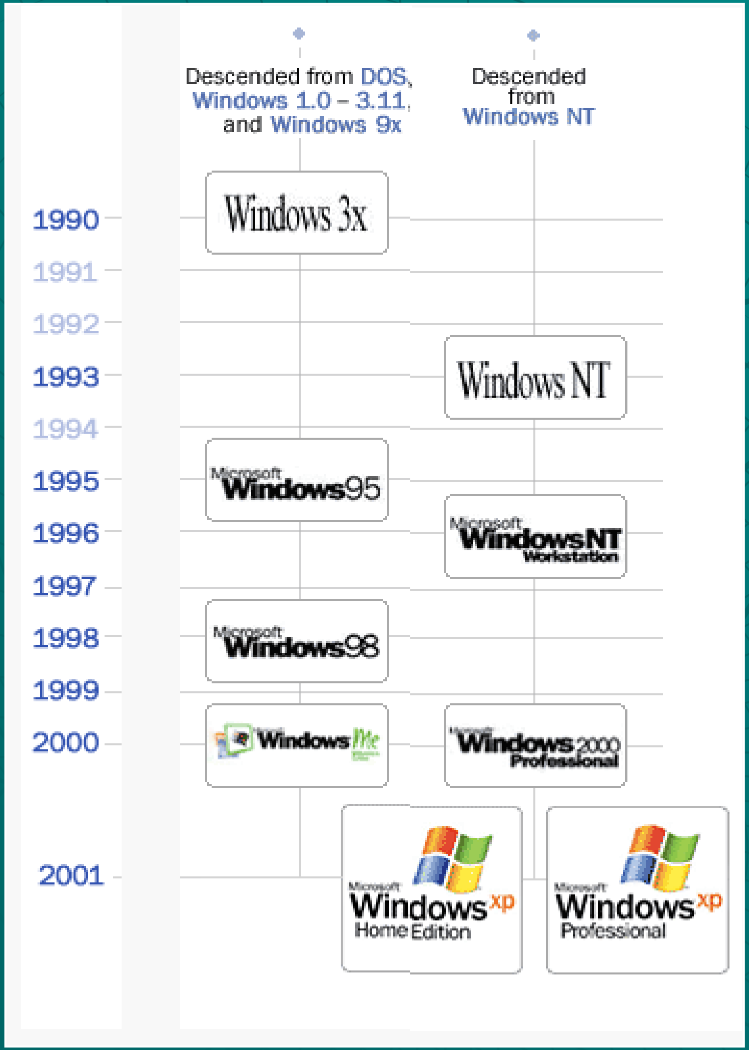
- Expensive computers.
- Limited CPU power and small memory.
- Computer’s time more valuable than programmer’s time.

New Reality

- *Human Complexity* – programmer’s time and effort – now far more important and expensive than the computer.
- *Human Complexity* is the bottleneck to achieving significant improvements in software productivity and quality.

Software evolves …

History of Microsoft Windows


Descended from DOS, Windows 1.0 – 3.11, and Windows 9x
Descended from Windows NT
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
Windows 3x
Windows NT
Microsoft Windows95
Microsoft Windows NT Workstation
Microsoft Windows98
Windows Me
Windows 2000 Professional
Microsoft Windows xp Home Edition
Microsoft Windows xp Professional

November 10, 1983, Microsoft announced Microsoft Windows®, an extension of the MS-DOS® operating system.

This marked the beginning of the graphical user interface (GUI) era at Microsoft.

→ Vista

Boundaries of our technologies?

Testing

- Testing - laborious and unreliable to guard against defects due to *unspecified behaviors*.
- A brute-force coverage of all paths is not possible – the number of execution paths is typically an astronomically large number.
- A test run implies traversing of one execution path at a time– defective paths may not be covered during the test runs.

Software Metrics

- The quantification problem is wide open.
- History tells us that it may be many years before we solve the problem.
- There were 35 different temperature scales before the Celsius.
- Starting with the earliest paper in 1968, we have more than 1000 software metrics – it will be a long time before we have metrics that we can agree on.

Documentation

- One guarantee we will always have – documentation of any useful software will never be complete.
- One guarantee we will never have – the documentation will always be consistent with the software.

Program Writing Technologies

- Examples: Programming languages, components and libraries, design patterns ..
- For writing new software: Program writing technologies have advanced a lot, but their focus is primarily on implementation rather than design and prototyping; they tend to obfuscate the domain knowledge.
- For evolving existing software: still a long way to go …

Software production

… a lot more besides writing it

Software – besides writing it

- Maintenance and evolution of software typically accounts for 65 to 70 percent of the total effort.
- Inspection and certification – critical for safety and mission critical applications.
- Reliability of software – the need to understand and address many complex issues.
- Preservation, propagation, and application of domain knowledge – critical for improving productivity and quality of software.

Software Evolution: A Classroom Experience

- In an operating system course project, a student spends:
  - *40* hours in identifying and understanding the relevant parts of code.
  - *2* hours in making the actual code changes to incorporate the specified functionality.
  - *10* hours in testing and debugging the code.
- The required code changes are small and the bugs are usually due to incomplete or incorrect understanding.

Inspection and Certification

- Coding standards:
  - The MISRA C for automotive industry
  - Ellemtel C++ for telecommunication industry
- DO-178B – the standard for safety-critical avionics software.

Software Reliability

- A hard problem because of the intrinsic domain complexity, the cost-performance tradeoffs, an ever broadening scope of applications, and the ever-changing implementation technologies.
- Critically important due to safety, security, and financial risks associated with software failures.

Domain Knowledge - The Key

- Imagine how difficult it would be to understand OS code without knowing the underlying OS design principles.
- Productivity and quality depends on:
  - *Preservation of knowledge* – withstanding the long software lifecycles and high turnover of personnel.
  - *Propagation of knowledge* – training new recruits.
  - *Application of knowledge* – bridging the gap between the domain knowledge and the code.

Program Reading

.. It deserves attention

Program Reading – A Critical Need

- Program Reading – analyzing, reasoning, and auditing software – overlooked by our educational system.
- We only have rudimentary program reading technologies – tools such as the open source *Cscope.*

Program Reading – It is not just for the developers

- Beyond developers, program reading can benefit managers.
- Get a survey of the software artifacts and their interrelationships – make informed decisions.
- Get an audit report – check compliance with quality requirements.

Program Reading

… It is not easy

Program reading is not easy

- Program reading is often difficult and challenging.
- Non-localized relationships between software artifacts.
- Complexity of semantic analysis.
- Lack of domain knowledge or the difficulty of applying it.

Cross-cutting Relationships and Complex Semantics

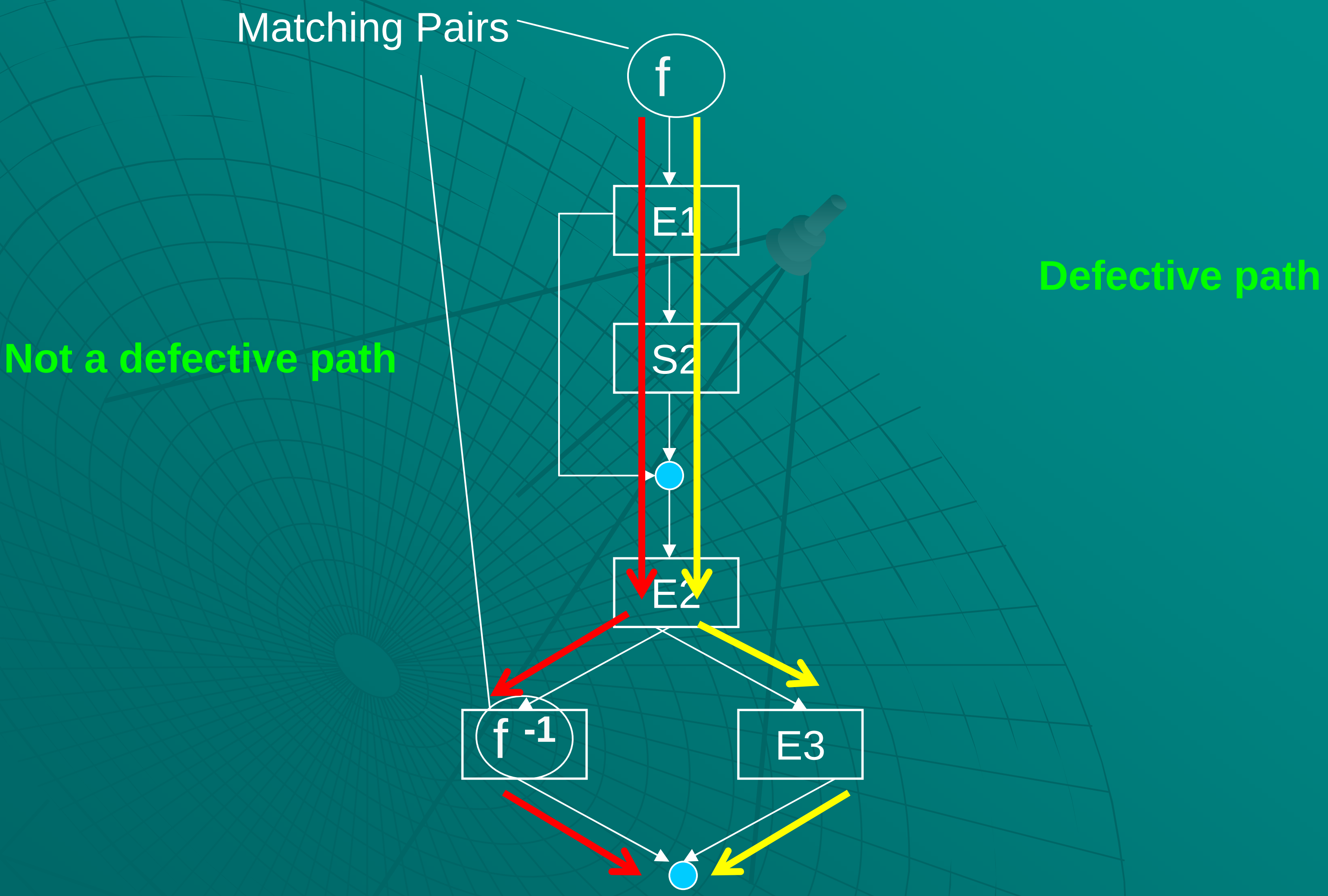
- As an example, we will present a series of semantically more complex *matching pair* (MP) *defects.*
- Basic pattern: to be correct, a program must have *matching pairs* of artifacts (e.g. parentheses) or events (e.g. locking and unlocking).

First Model of MP Defects

- Must have matching pairs of syntactic patterns – for example, parentheses.
- Must analyze *syntax* to find defects.

Second Model of MP Defects

- Must have matching pairs *f* and f^{-1} along each *execution path*.
- Must analyze the *control flow* to find defects.


Matching Pairs
f
E1
S2
E2
f -1
E3
Not a defective path
Defective path

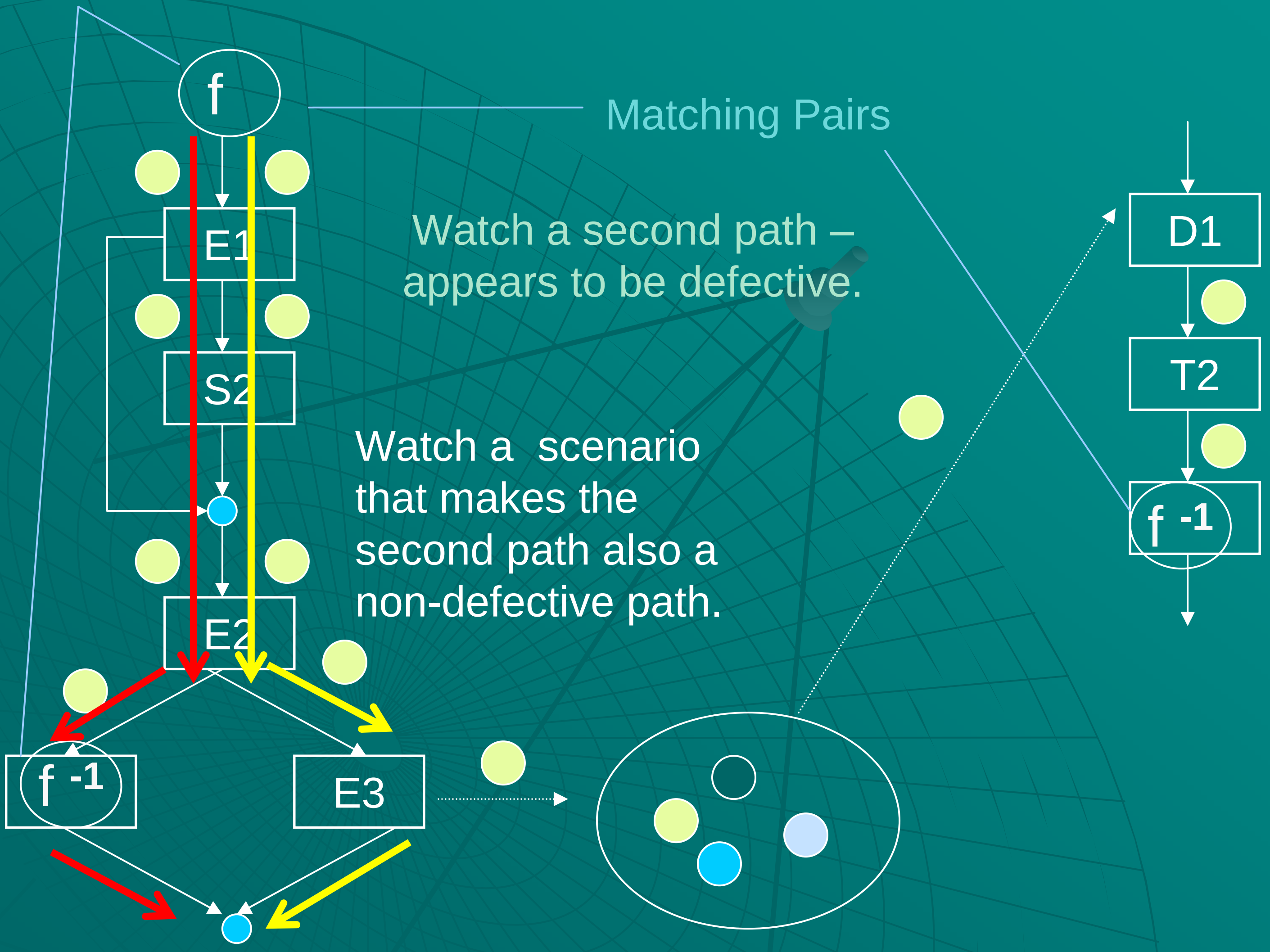
Third Model of MP Defects

- Must have the same *id* associated with *f and f* $^{-1}$.
- The *id* gets passed through tokens.
- Must analyze the *control and data flow* to find defects.

Fourth Model of MP Defects

- *f* and *f* $^{-1}$ may occur on disconnected paths due to concurrency and interrupt processing.
- Must analyze the *control flow*, *data flow* and *disconnected execution paths*.

Watch one non-defective path.



Matching Pairs

Matching Pairs

Watch a second path – appears to be defective.

Watch a scenario that makes the second path also a non-defective path.

disconnected execution path

Shared pool of tokens

Observations

- Compilers cannot catch the defects beyond the first model, because they are limited by the kind of static program analysis they can perform.
- The static program analysis required for an accurate solution for the second and third model is known to be intractable (Rice-Myhill-Shapiro theorem, 1953).

Program Reading Technologies

… Wave of the future

Program Reading Technologies

- Goal: Provide tools to assist with the reading of complex programs.
- Tools will address specific needs: program understanding, smart testing, defect analysis, impact analysis, inspection, auditing …

“Assigning more programmers to a project running behind schedule will make it even later.”

COMPUTER ARCHITECTURE
COMPUTER ARCHITECTURE
AUTOMATIC DATA PROCESSING
A Programming Language
Numquam Incertus, Semper Apertus
Never uncertain, Always open (to suggestion)
for your consideration...
Comp 236 S 96 Lectures

Fred Brooks – A Legend in Computing

- Architect of the IBM 360 OS.
- Author of the classic - *The Mythical Man-Month: Essays on Software Engineering.*
- National Medal of Technology (1985).
- A.M. Turing Award, Association for Computing Machinery (1999).

Tools to amplify human intelligence

Fredrick Brooks:

> "… IA > AI, that is, that intelligence amplifying systems can, at any given level of available systems technology, beat AI systems. That is, a machine and a mind can beat a mind-imitating machine working by itself."

Program Reading Technologies

… Work at ISU and EnSoft

Knowledge-Centric Software (KCS) Tools

- Goal: Enable program reading to preserve, propagate, or apply domain knowledge.
- KCS tools automate execution of intelligent program reading strategies:
  - User designs the strategies.
  - Tools help in executing the strategies by automating extraction and modeling of information about software artifacts and their relationships.

Query-Model-Refine (QMR) Approach

- A natural way to *amplify human intelligence* by assisting in:
  - Retrieval of information by analyzing software.
  - Generation of visual models from the retrieved information.
  - Refinement of the models to manage complexity.

Tools Technology

- eXtensible Common Intermediate Language (XCIL): a language for referring to program artifacts – makes it easier to develop tools for different languages (Ada, COBOL, C, C++, FORTRAN, and Java).
- eXtensible Pattern Specification Language (XPSL): a language for referring to relations between program artifacts – makes it easier to develop tools for different domains (control software, systems software, numerical software, business software).

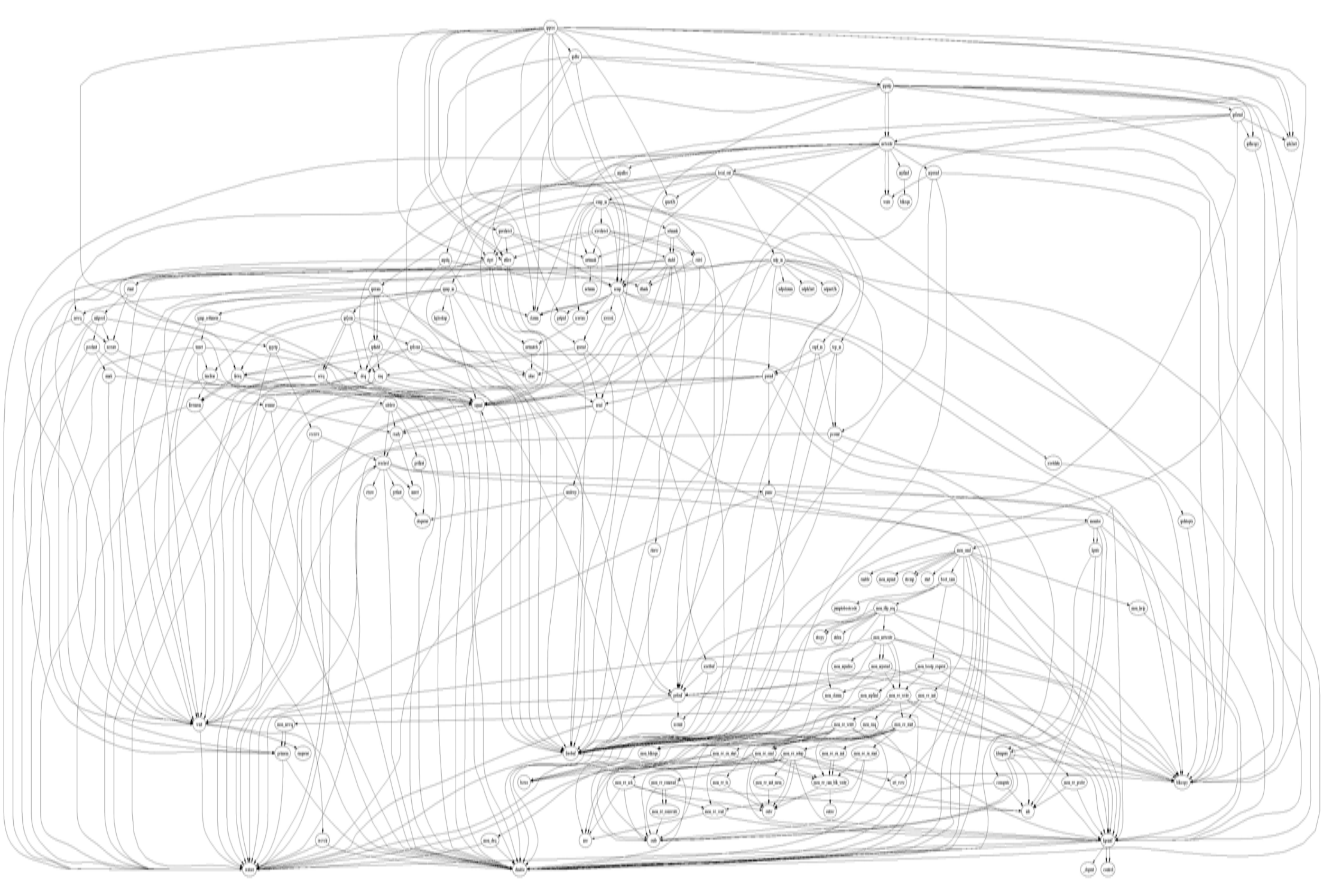
Program Reading Tools

… A Trailer

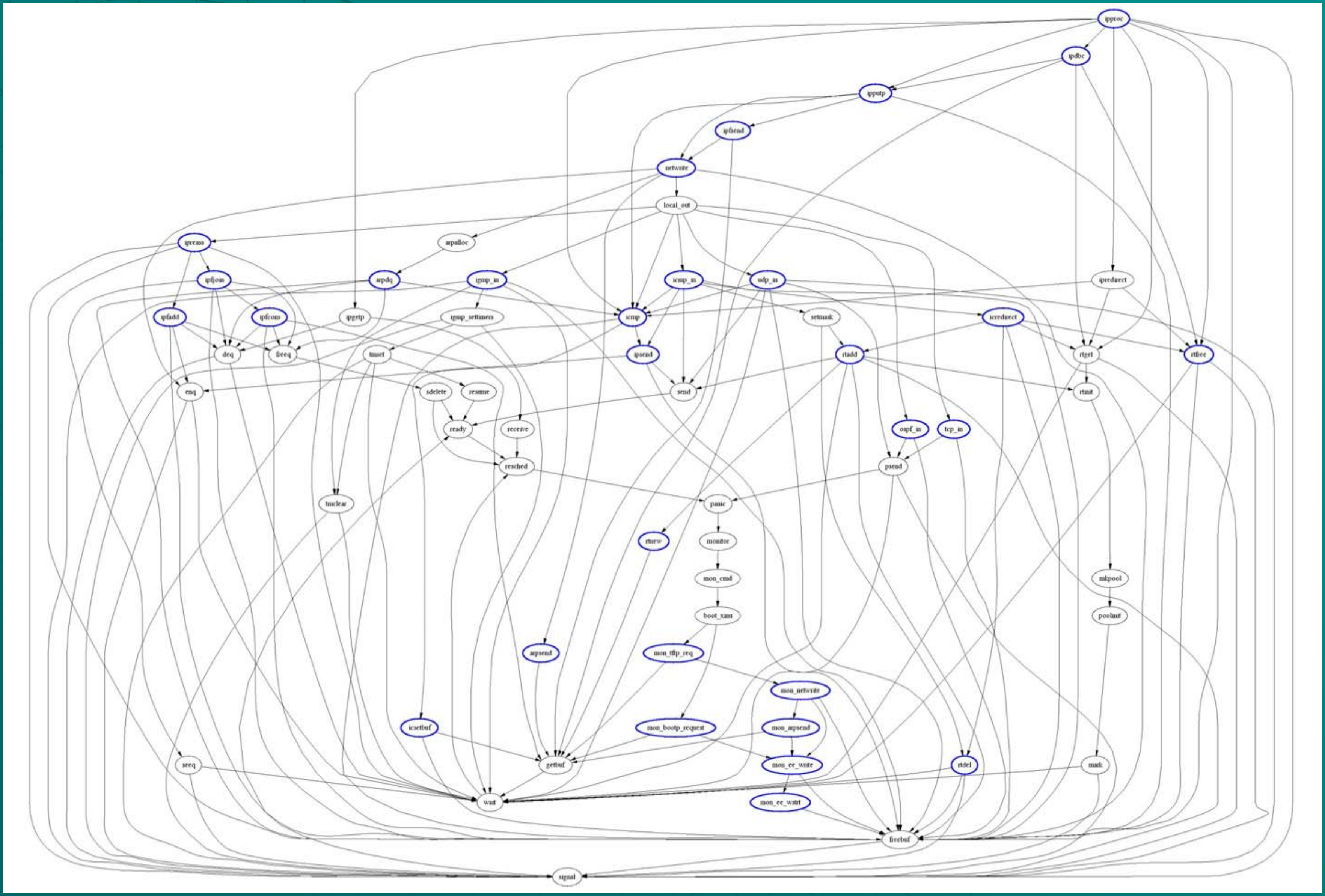
An Example of Architecture Extraction

- Extracting the architecture of XINU networking code.
- A model and two successive refinements:
  - View I: Call graph (CG) as a model.
  - View II: CG after one refinement.
  - View III: CG after the second refinement.

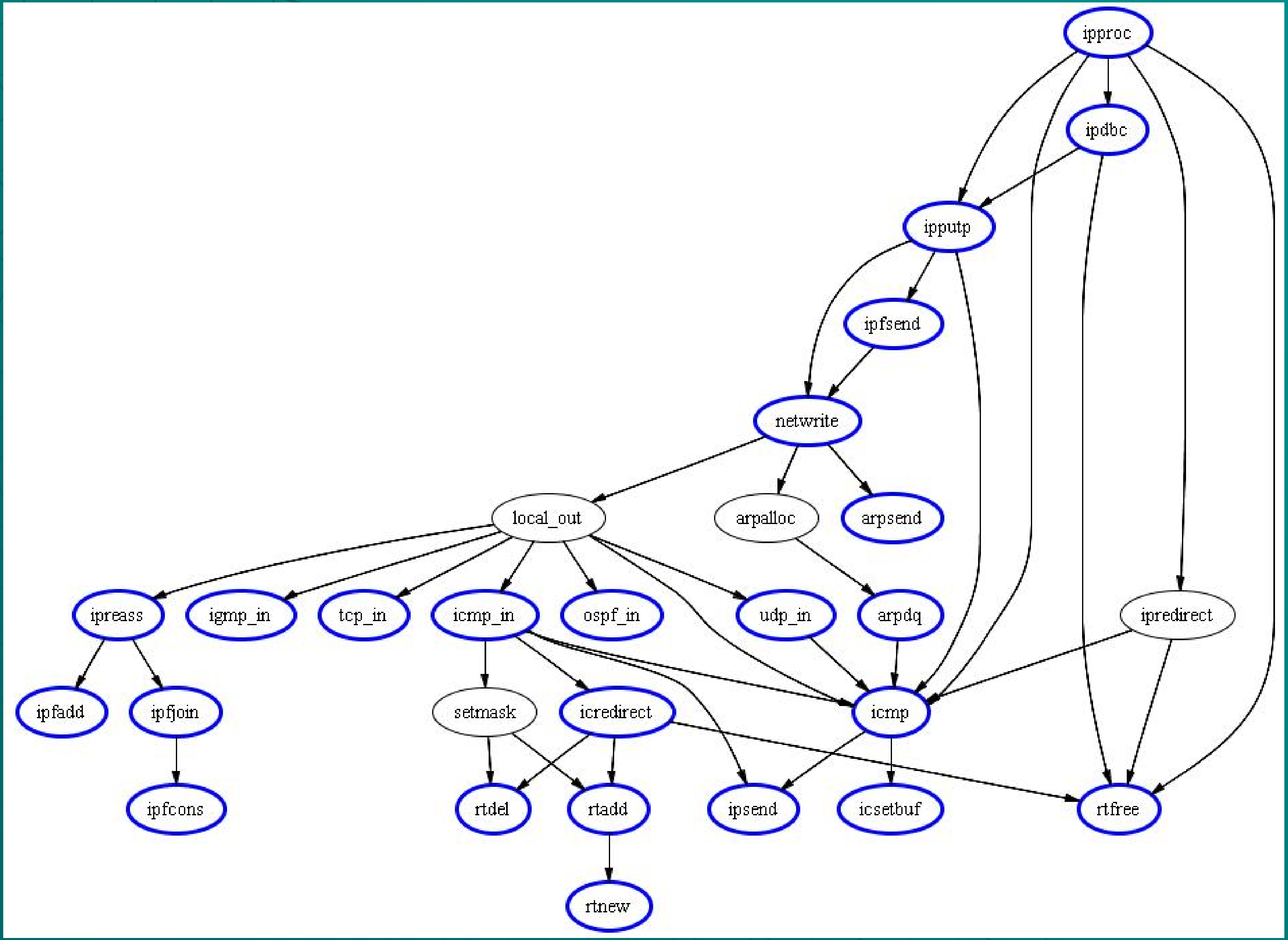
The Call Graph Model



After one refinement



After the second refinement

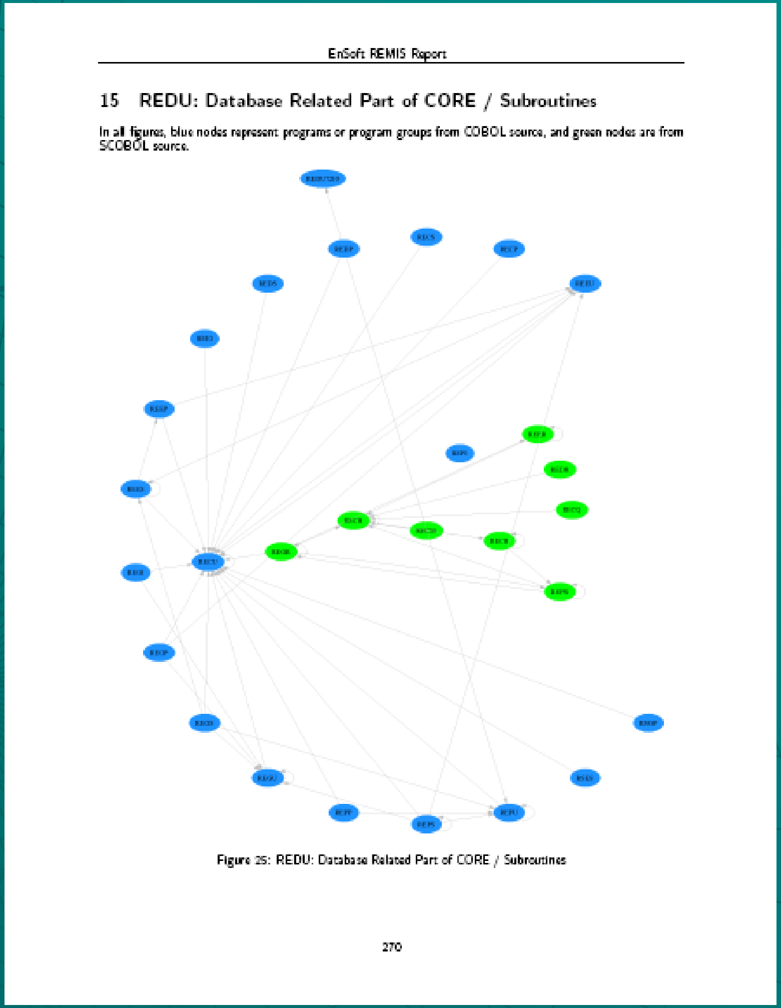


Program Reading Tools

… demos and discussions

Program Reading Tools From EnSoft

Total Insight – A COBOL Tool

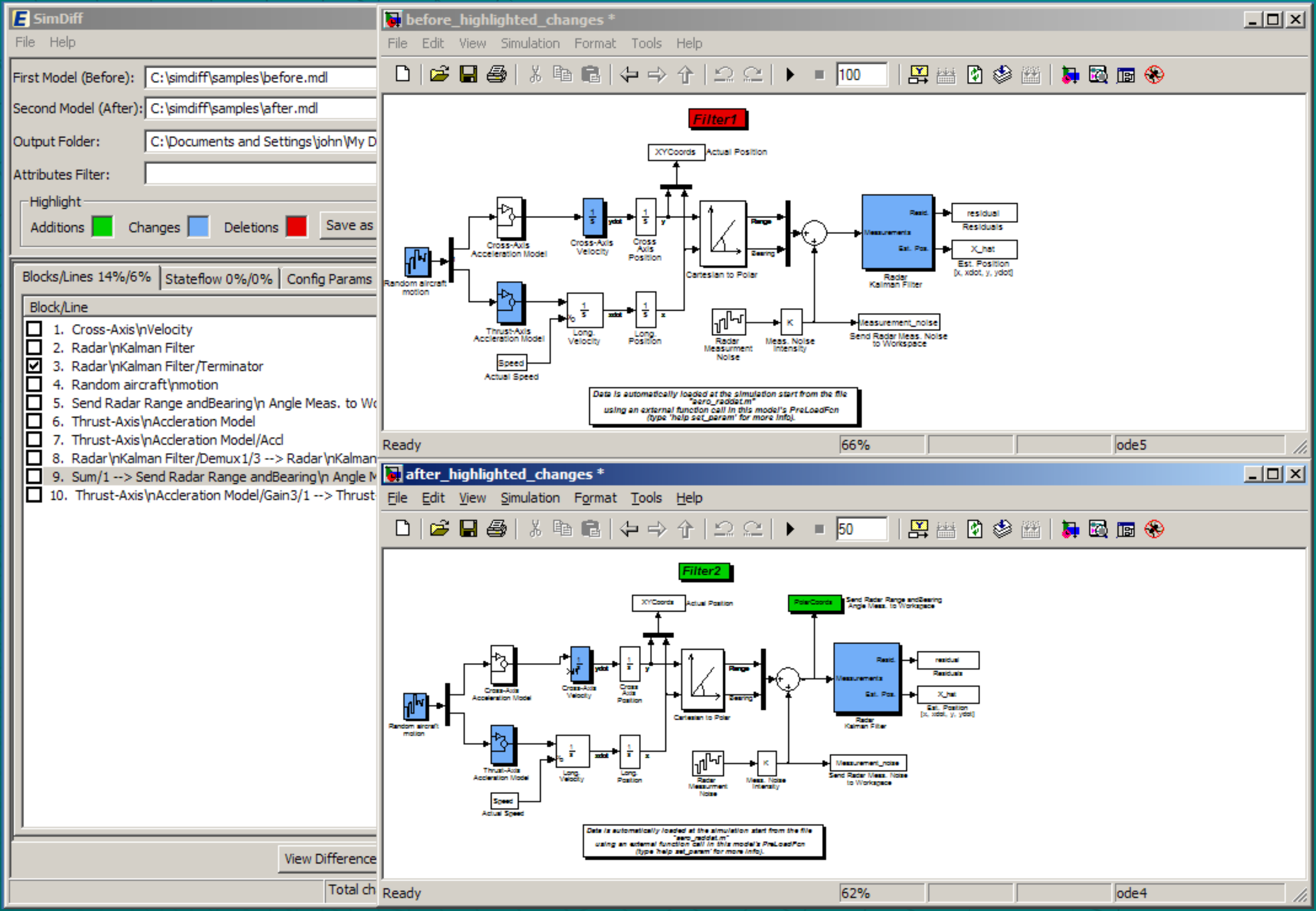


15 REDU: Database Related Part of CORE / Subroutines

In all figures, blue nodes represent programs or program groups from COBOL source, and green nodes are from SCOBOL source.

Figure 25: REDU: Database Related Part of CORE / Subroutines

SimDiff – A Model Differencing Tool



First Model (Before): C:\simdiff\samples\before.mdl

Second Model (After): C:\simdiff\samples\after.mdl

Output Folder: C:\Documents and Settings\john\My D

Attributes Filter:

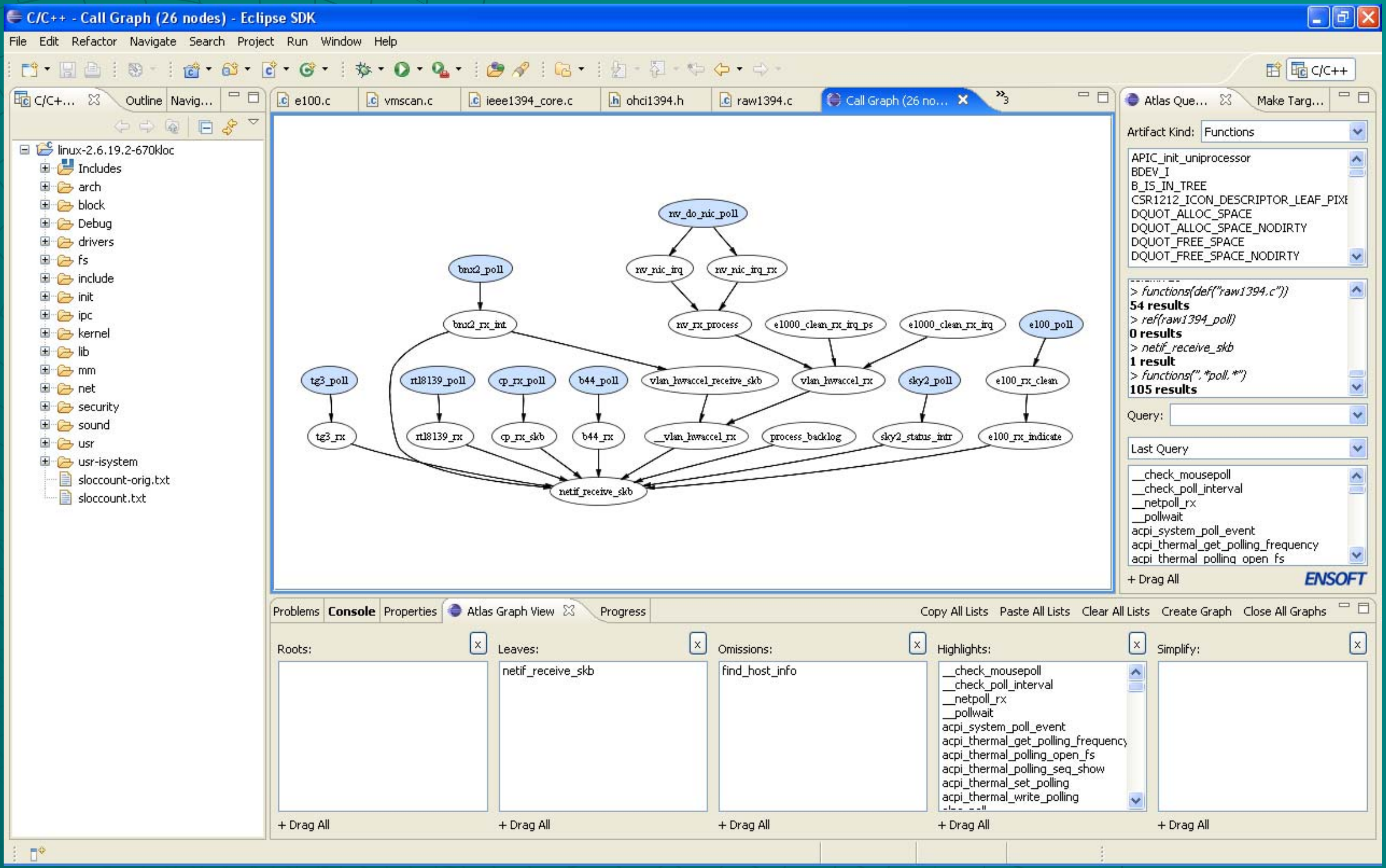
Additions 🟩 Changes 🟦 Deletions 🟥

□ 1. Cross-Axis\nVelocity
□ 2. Radar\nKalman Filter
☑ 3. Radar\nKalman Filter/Terminator
□ 4. Random aircraft\nmotion
□ 5. Send Radar Range andBearing\n Angle Meas. to Wc
□ 6. Thrust-Axis\nAccleration Model
□ 7. Thrust-Axis\nAccleration Model/Accl
□ 8. Radar\nKalman Filter/Demux1/3 --> Radar\nKalman
□ 9. Sum/1 --> Send Radar Range andBearing\n Angle M
□ 10. Thrust-Axis\nAccleration Model/Gain3/1 --> Thrust

Data is automatically loaded at the simulation start from the file "aero_raddat.m" using an external function call in this model's PreLoadFcn (type 'help set_param' for more info).

Data is automatically loaded at the simulation start from the file "aero_raddat.m" using an external function call in this model's PreLoadFcn (type 'help set_param' for more info).

Atlas – A Program Mapping Tool



Artifact Kind: Functions

APIC_init_uniprocessor
BDEV_I
B_IS_IN_TREE
CSR1212_ICON_DESCRIPTOR_LEAF_PIXE
DQUOT_ALLOC_SPACE
DQUOT_ALLOC_SPACE_NODIRTY
DQUOT_FREE_SPACE
DQUOT_FREE_SPACE_NODIRTY

```
> functions(def("raw1394.c"))
54 results
> ref(raw1394_poll)
0 results
> netif_receive_skb
1 result
> functions(".*poll.*")
105 results
```

Query:

Last Query

__check_mousepoll
__check_poll_interval
__netpoll_rx
__pollwait
acpi_system_poll_event
acpi_thermal_get_polling_frequency
acpi_thermal_polling_open_fs

+ Drag All

Roots:

+ Drag All

Leaves: netif_receive_skb

+ Drag All

Omissions: find_host_info

+ Drag All

Highlights:

__check_mousepoll
__check_poll_interval
__netpoll_rx
__pollwait
acpi_system_poll_event
acpi_thermal_get_polling_frequency
acpi_thermal_polling_open_fs
acpi_thermal_polling_seq_show
acpi_thermal_set_polling
acpi_thermal_write_polling

+ Drag All

Simplify:

+ Drag All

References

- MISRA C Coding Standard:
  - http://www.knosof.co.uk/misracom.html
  - http://www.misra.org.uk/
- Ellemtel C++ Coding Standard:
  - http://www.doc.ic.ac.uk/lab/cplus/c++.rules/
- Avionics software standard:
  - http://en.wikipedia.org/wiki/DO-178B
- Cscope:
  - http://cscope.sourceforge.net/
- Fredrick Brooks, *The Computer Scientist as Toolsmith*,
  - http://www.cs.unc.edu/~brooks/Toolsmith-CACM.pdf
- Earliest paper on software complexity:
  - Rubey, R.J.; Hartwick, R.D.: Quantitative Measurement Program Quality. ACM, National Computer Conference pp. 671-677, 1968.
- Knowledge-Centric Software Research Laboratory at ISU
  - http://dirac.ece.iastate.edu/sec/
- EnSoft Corp.
  - http://www.ensoftcorp.com/