“The fault detectability domain: Attack in depth"

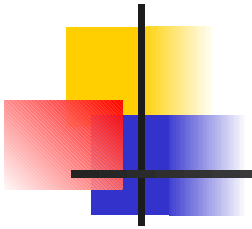
by

Les Hatton

The Computing Laboratory, University of Kent
L.Hatton@ukc.ac.uk, lesh@oakcomp.co.uk

Version 1.2: 17/Oct/2002

Overview



- Principles
- Technical input
- Implementation policy
- Architecture
- Status
- Future direction

Technologies

The following technologies are available for detecting defects

- Direct interpretation (DI)

  A program is checked for strict linguistic conformance and freedom from a wide class of related faults.

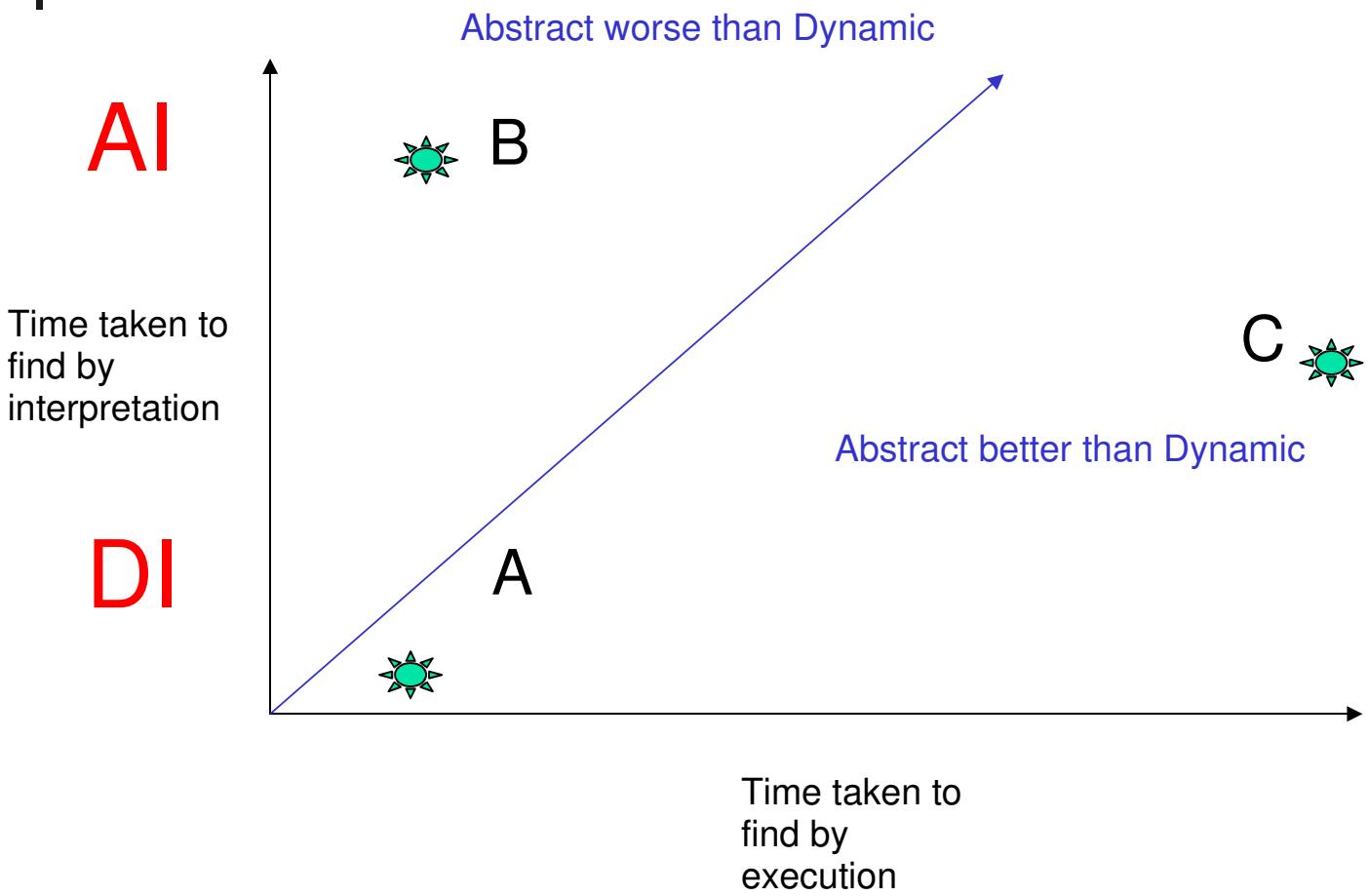
- Abstract interpretation (AI)

  Linear relationships are extracted or approximated from a program and linear algebra used to infer equalities or inequalities which could lead to failure.

- Abstract execution (AE)

  The program is run in parallel with the defined abstract model of computation. Failing faults are detected directly.

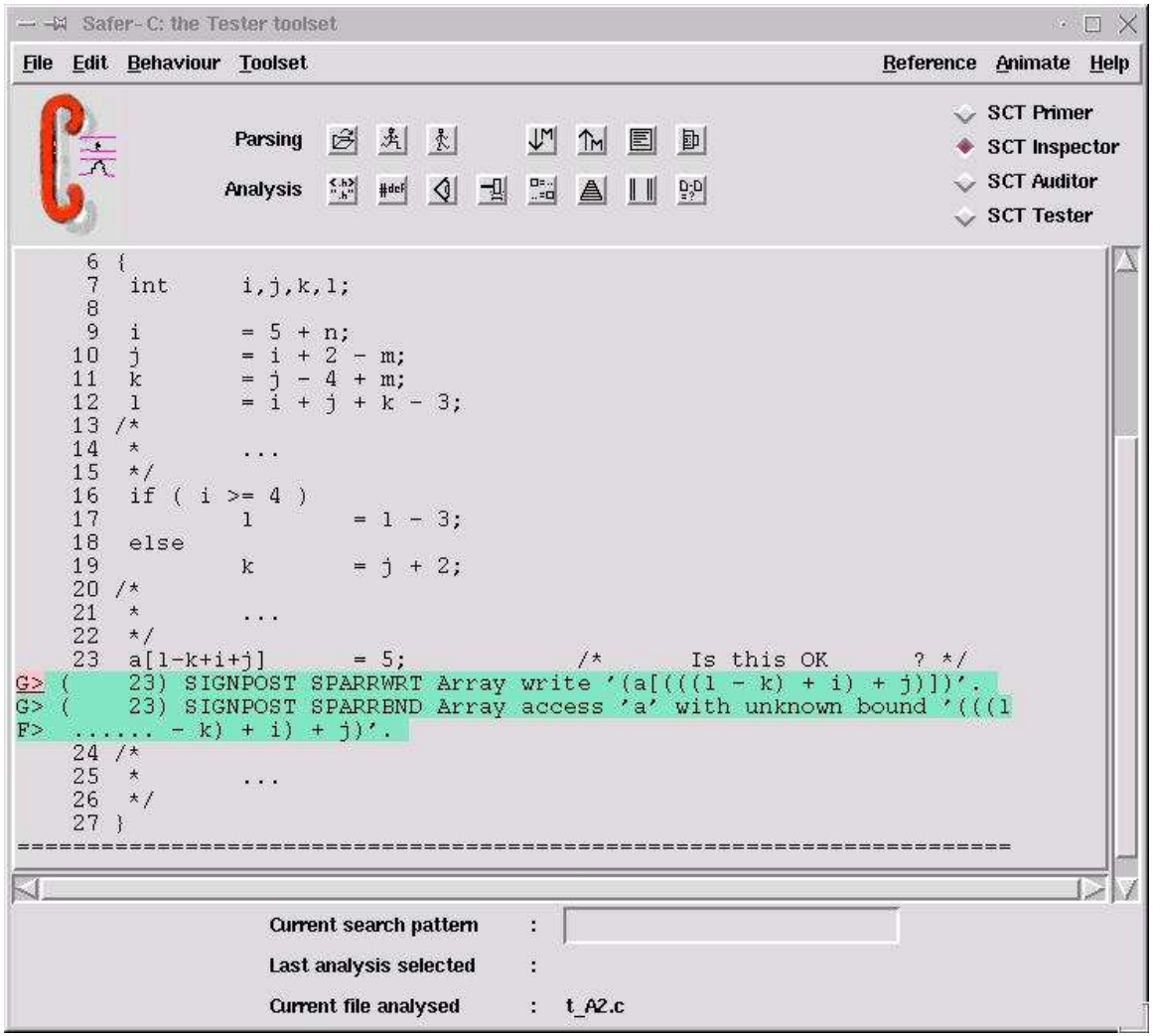
The detectability domain



Direct interpretation - single component

Advantage
Very fast

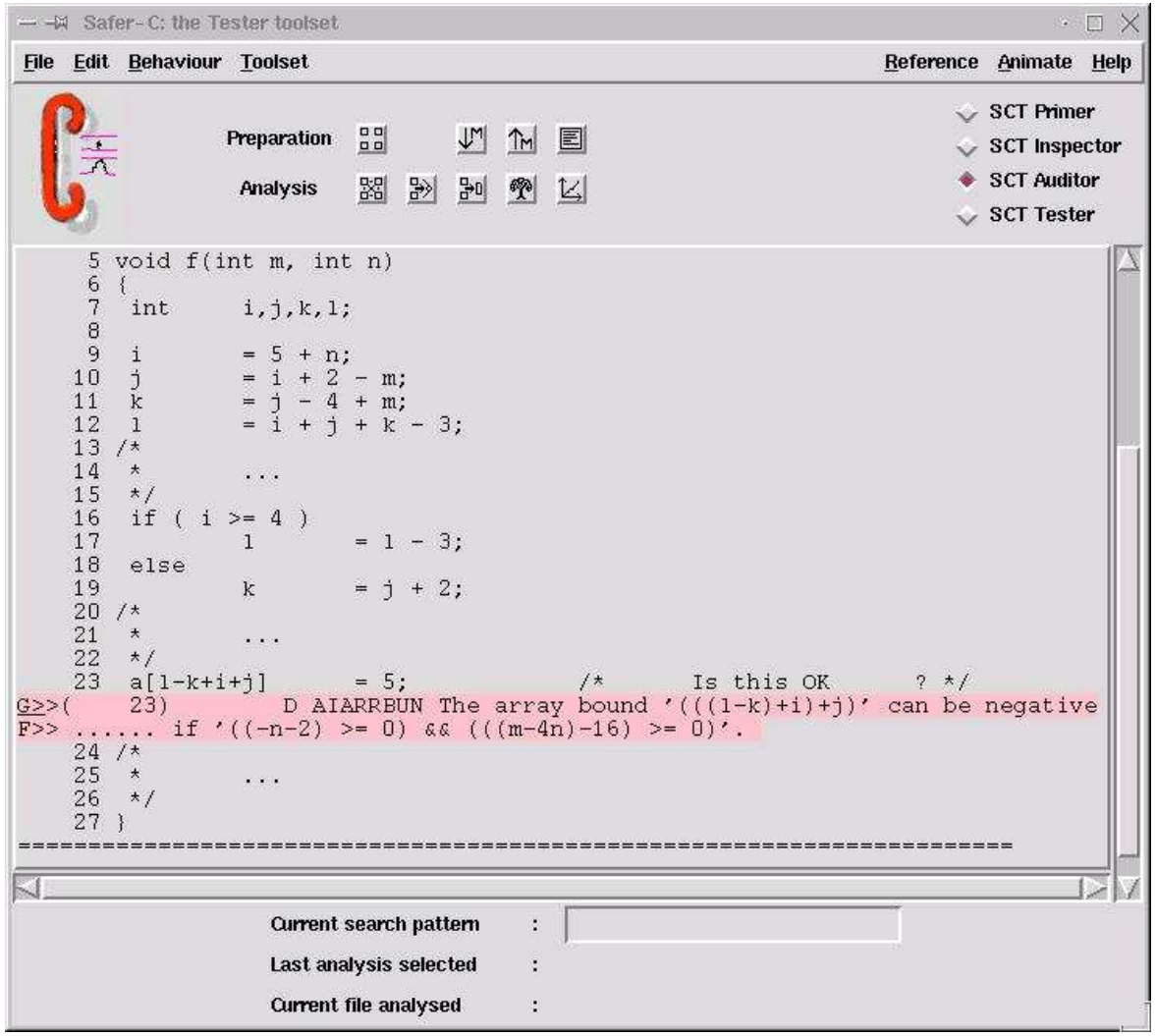
Disadvantage
Onus on user to Inspect.



Abstract interpretation - single component

Advantage
Almost as fast

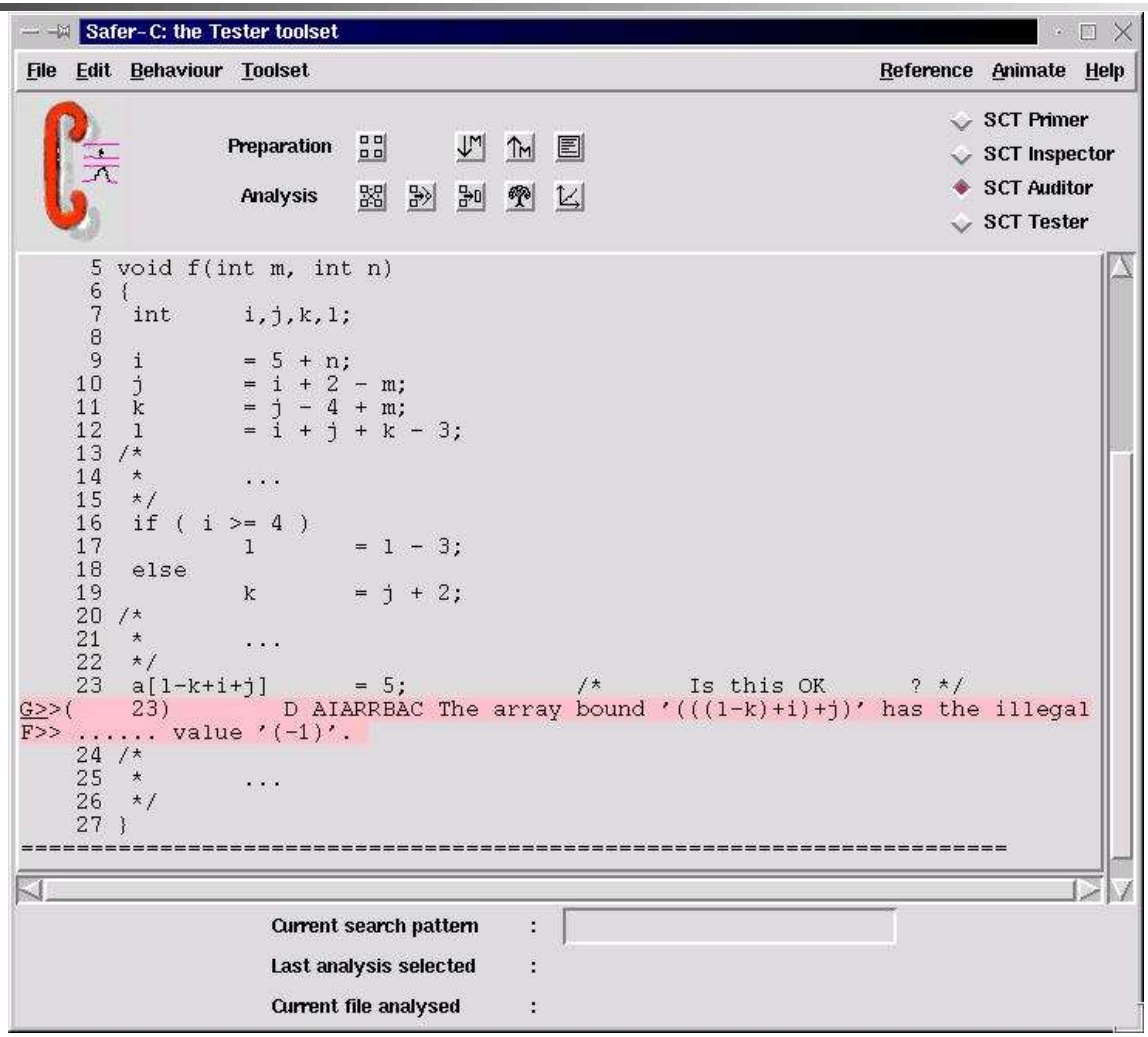
Disadvantage
onus on user to inspect but more information provided.



Abstract interpretation - system wide

Advantage
Explicit detection

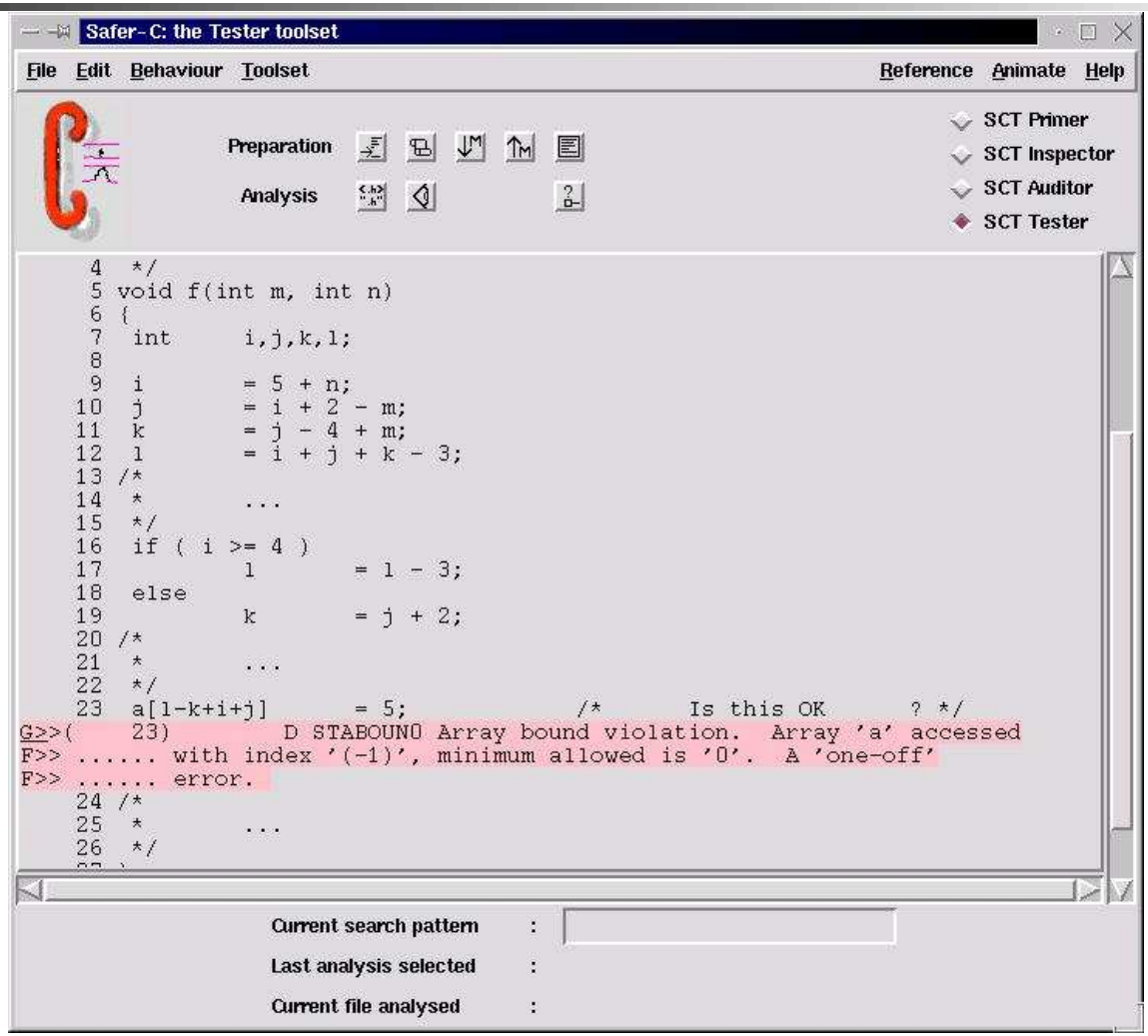
Disadvantage
Took many hours



Abstract execution - system wide

Advantage
Explicit detection

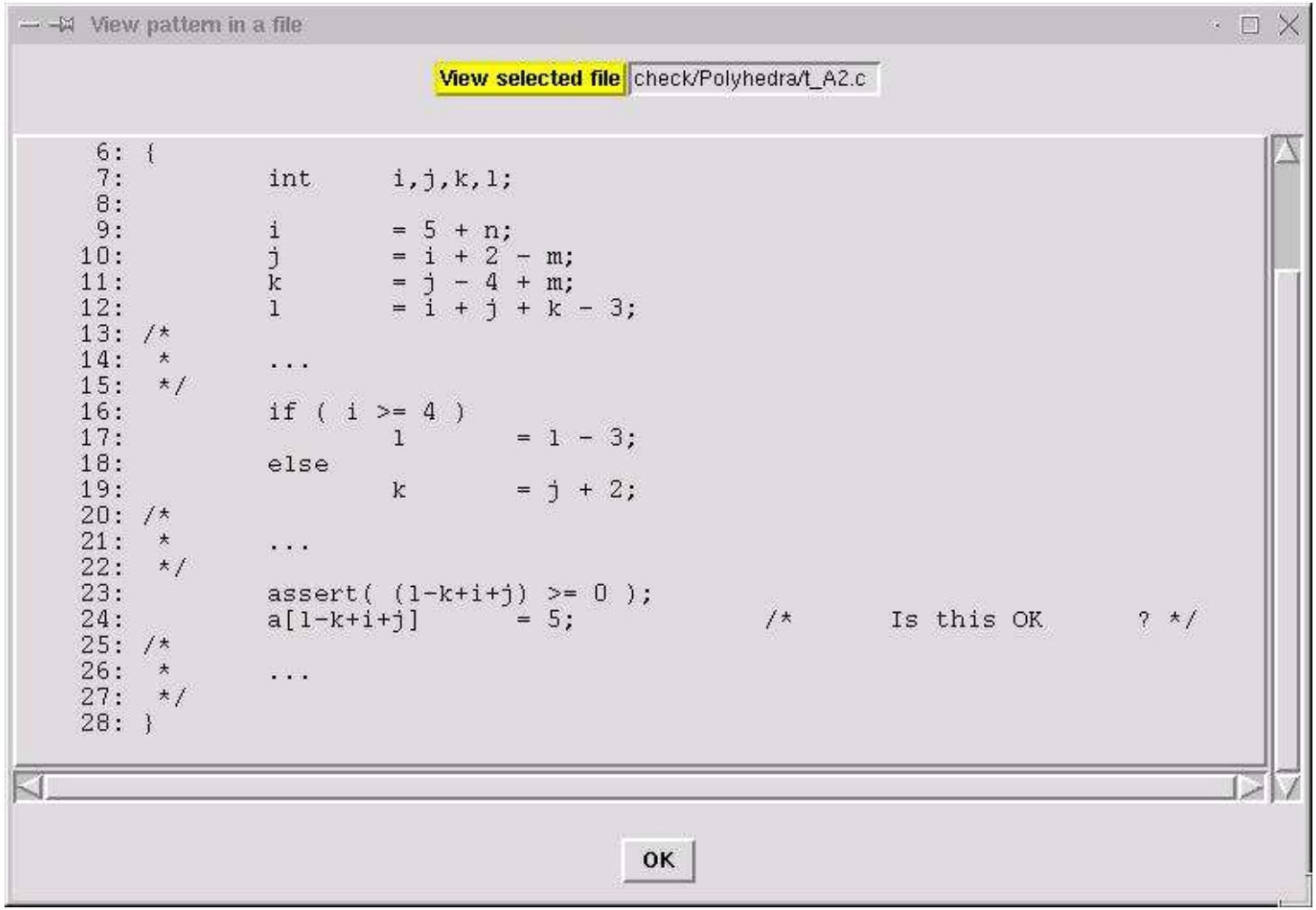
Disadvantage
Much slower than normal real time
But faster than Abstract Interpretation



Direct execution – assertion defence

Advantage
Very fast

Disadvantage
Programmer must annotate.

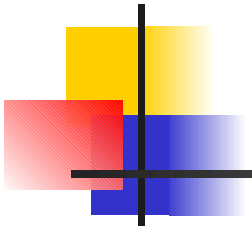


Technical input

The following sources were used for version 1.1:-

- The April 1998 MISRA document, ISBN 0-9524156-9-0
- The (undated) Technical Clarification – Draft one, authored by Gavin and Paul
- The existing conformance suites provided by Andrew Burnard, Derek Jones and Les Hatton
- Ongoing input from the steering committee

Implementation policy



The following simple policy was used:-

- Hierarchy of wording
  - Bold face part of rule assumed normative overriding any commentary
  - Commentary used if bold face ambiguous
  - False positives favoured if commentary ambiguous; query files produced; rationale modified.
- No dependence on file inclusion
- Required rules done first

Architecture: Goals

Goals:-

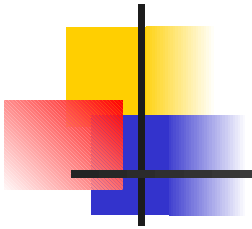
- Orthogonality, (no crosstalk but *not* 1 item 1 file). I failed :-(
- Portability. (The architecture should run on any platform)
- Verifiability. There must be a way of detecting whether the source of a rule was altered. There must also be no dependence on inclusion, so parts of the C standard headers had to be constructed.

Architecture: Construction

Construction:-

- Perl was used for the engine. This should run without change on Windows, Linux, Unix and so on although are still two shell scripts in 1.1.
- An automatic checksum is inserted into the conformance suite on a file by file basis so that source change can be detected.

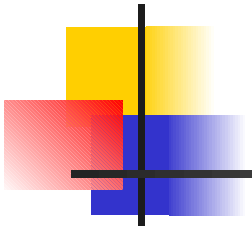
Architecture: Deliverables



All deliverables are open source, released under the GPL:-

- Readme + Rationale (html) + GPL licence
- MISRA conformance suite, (completely self-contained)
  - Every rule has a positive file (must detect) and a negative file (must not detect)
  - Some rules have query files (not part of conformance but may be in future)
- Conformance suite architecture

Status



Version 1.1c available now:-

- 497 test cases for 45 rules in 90 files. (40 days work so far in 46 CRs). (Milestone 15-10-2001, 23:59:00)

Version 1.1...

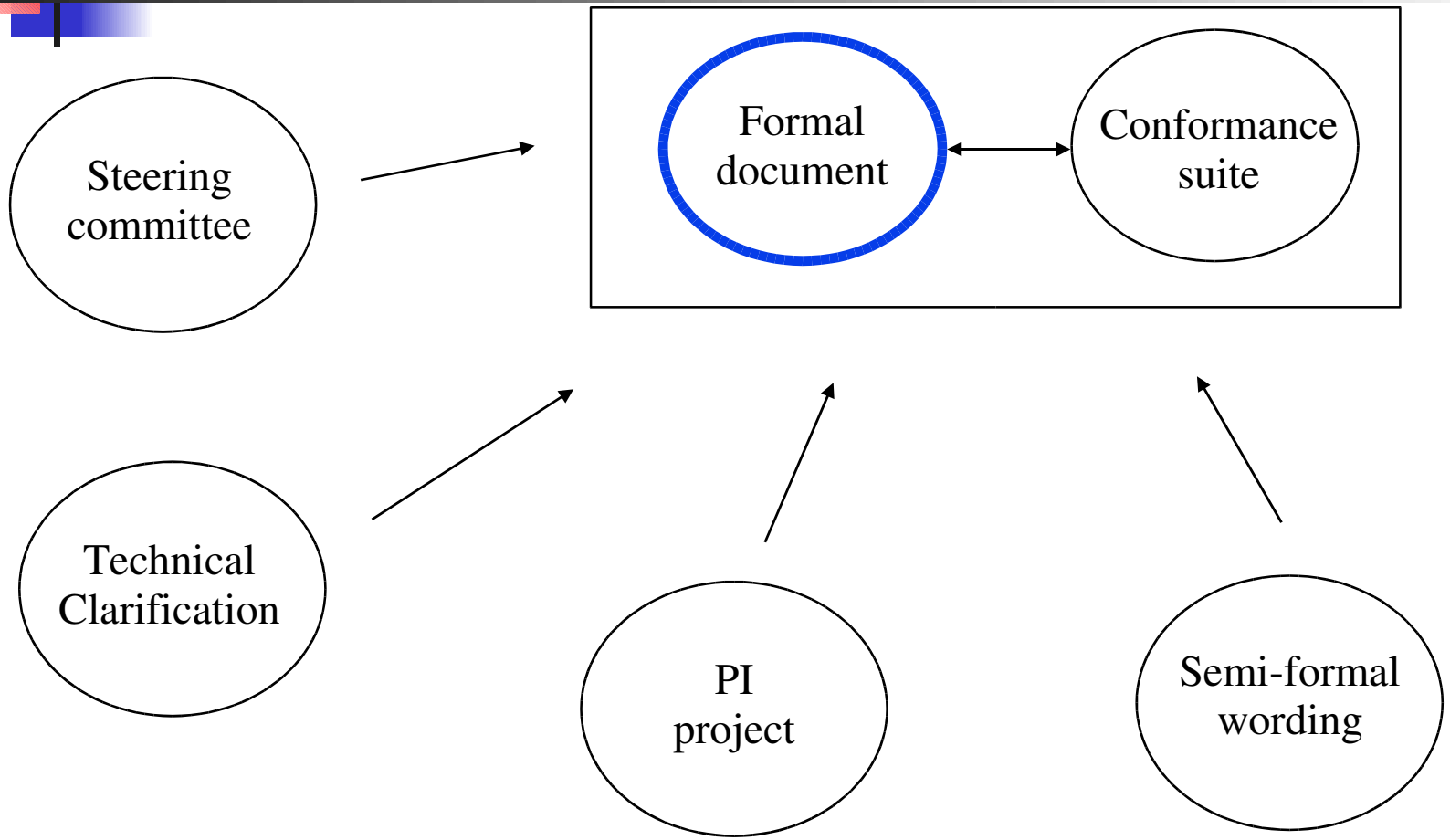
- Remaining required rules a few at a time

Version 2.0

- Advisory rules also a few at a time

Later versions to assimilate revised wording as it develops

Future direction: Sources of change



Availability

Download sites:-

- The conformance suite itself is available at the UKC national Safer Language Subsets project site:-

  http://www.cs.ukc.ac.uk/national/SLS