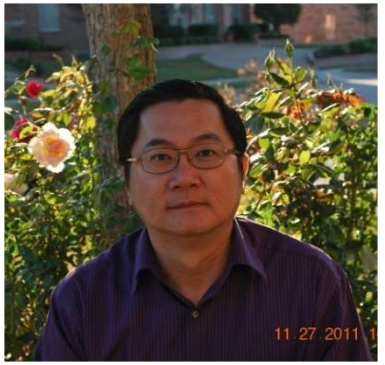
Coverage Testing SDL Models

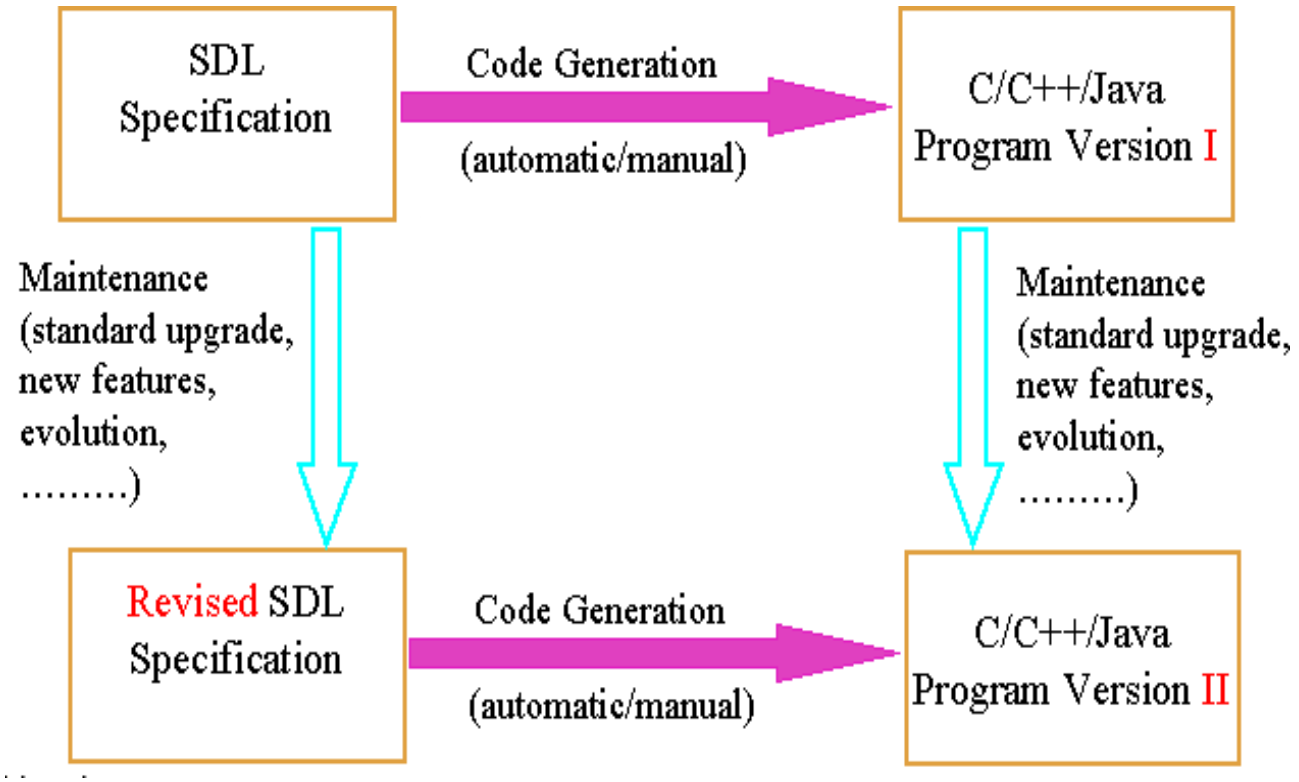
W. Eric Wong
Department of Computer Science
The University of Texas at Dallas
ewong@utdallas.edu
http://www.utdallas.edu/~ewong

Speaker Biographical Sketch

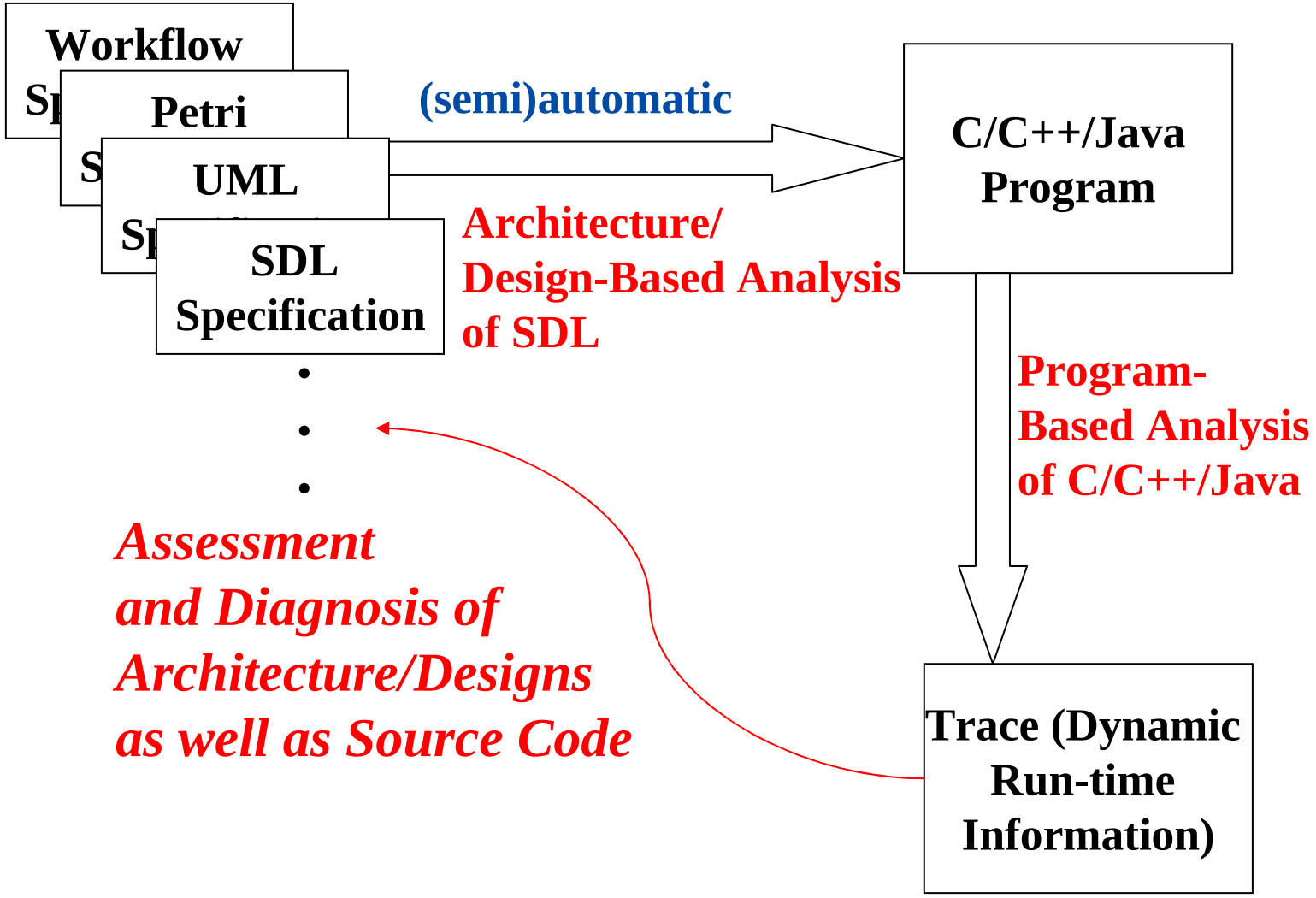


- Professor & Director of International Outreach
  Department of Computer Science
  University of Texas at Dallas
- Guest Researcher
  Computer Security Division
  National Institute of Standards and Technology (NIST)
- Vice President, IEEE Reliability Society
- Secretary, ACM SIGAPP (Special Interest Group on Applied Computing)
- Principal Investigator, NSF TUES (Transforming Undergraduate Education in Science, Technology, Engineering and Mathematics) Project
  - *Incorporating Software Testing into Multiple Computer Science and Software Engineering Undergraduate Courses*
- Founder & Steering Committee co-Chair for the SERE conference (*IEEE International Conference on Software Security and Reliability*) (http://paris.utdallas.edu/sere13)

Software Development

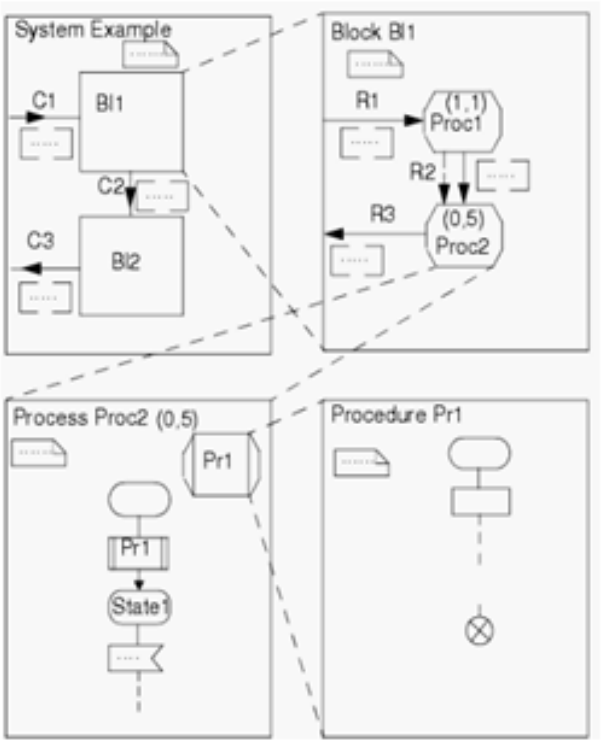


Our Vision

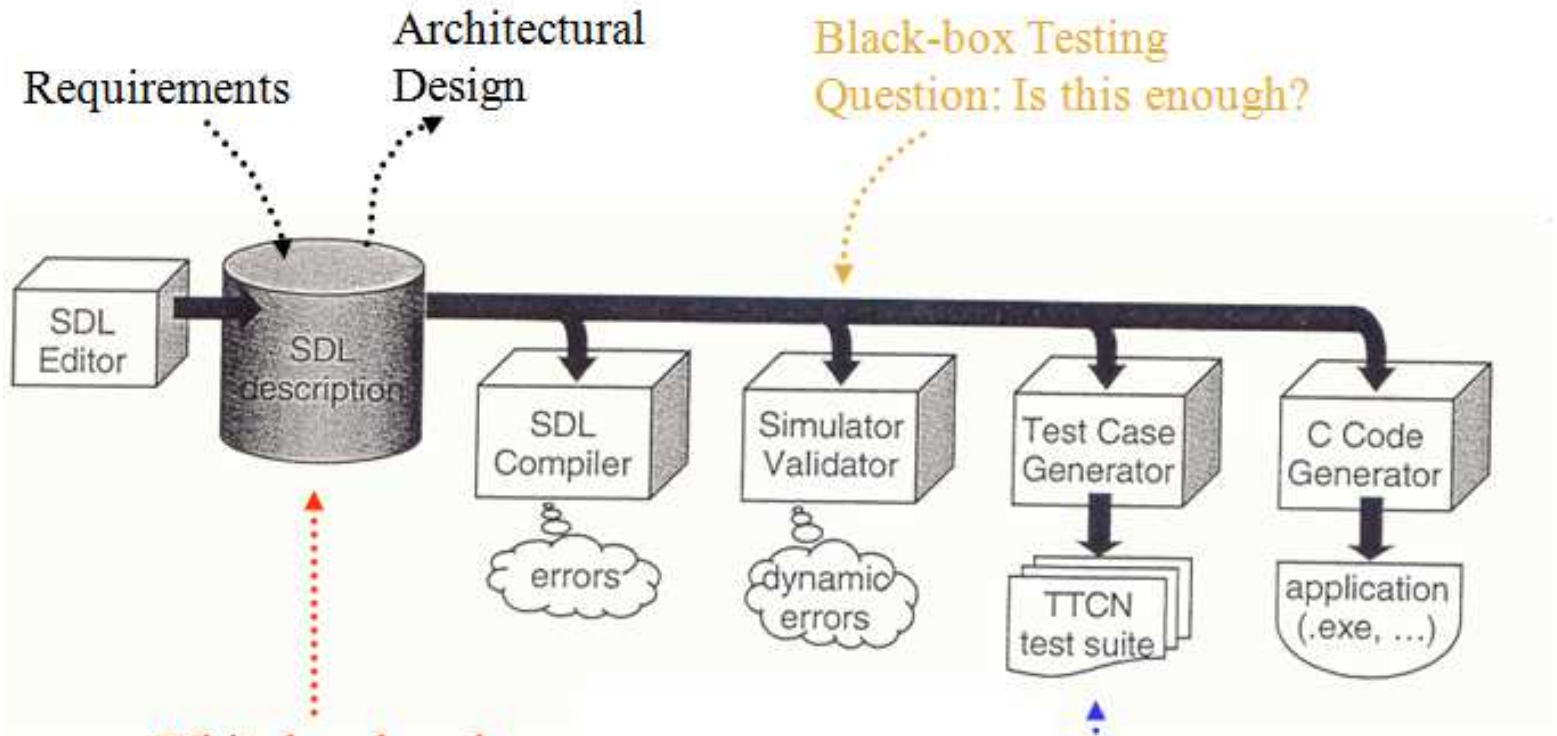

Workflow
Petri
UML
SDL
Specification
(semi)automatic
Architecture/
Design-Based Analysis
of SDL
C/C++/Java
Program
Program-
Based Analysis
of C/C++/Java
Trace (Dynamic
Run-time
Information)
Assessment
and Diagnosis of
Architecture/Designs
as well as Source Code

Software Architecture Design in SDL

- SDL (Specification and Description Language) is an object-oriented, formal language for designing *complex*, *real-time*, and *communicating* systems
  - Visit http://www.sdl-forum.org for more details
- The architectural design of a software system in SDL can be viewed as a collection of blocks and processes communicating with each other by exchanging signals through channels.
  - An SDL specification provides *a process view* of a system's architectural design



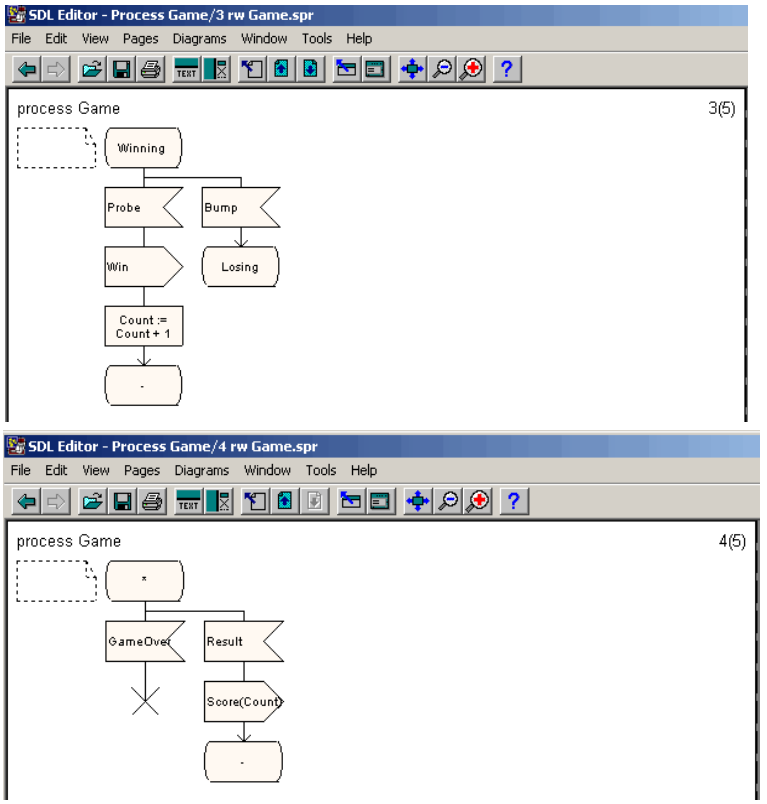
Life of an SDL Specification



Graphical and Textual Representations

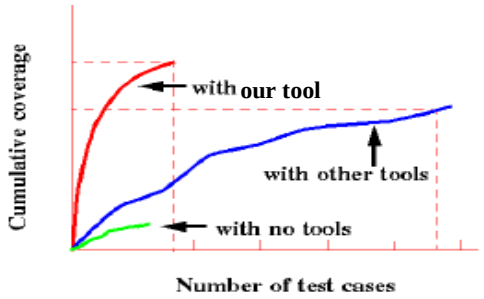
- The graphical representation is called GR and the textual representation is called PR (*Phrase Representation*).
  - Automatic translation between GR and PR can be done

```
process Game;
dcl
  Count Integer;

start;
    task Count := 0;
    nextstate Losing;

state Losing;
  input Probe;
    output Lose;
    task Count := Count - 1;
    nextstate -;
  input Bump;
    nextstate Winning;
endstate;

state Winning;
  input Bump;
    nextstate Losing;
  input Probe;
    output Win;
    task Count := Count + 1;
    nextstate -;
endstate;

state * ;
  input Result;
    output Score(Count);
    nextstate -;
  input GameOver;
    stop;
endstate;
endprocess Game;
```



Coverage Testing SDL Specifications (1)

- The textual representation of SDL specifications can be viewed as "*programs*" *in a specification and description language*, just like programs in C.
  - All the testing methods applied to C programs, including random testing and functional testing (both are black box oriented) as well as control flow-based and data flow-based white box coverage testing, can also be applied to SDL specifications.
  - How much of the design specification is currently tested?
  - What is missing?
  - Need help in creating tests?

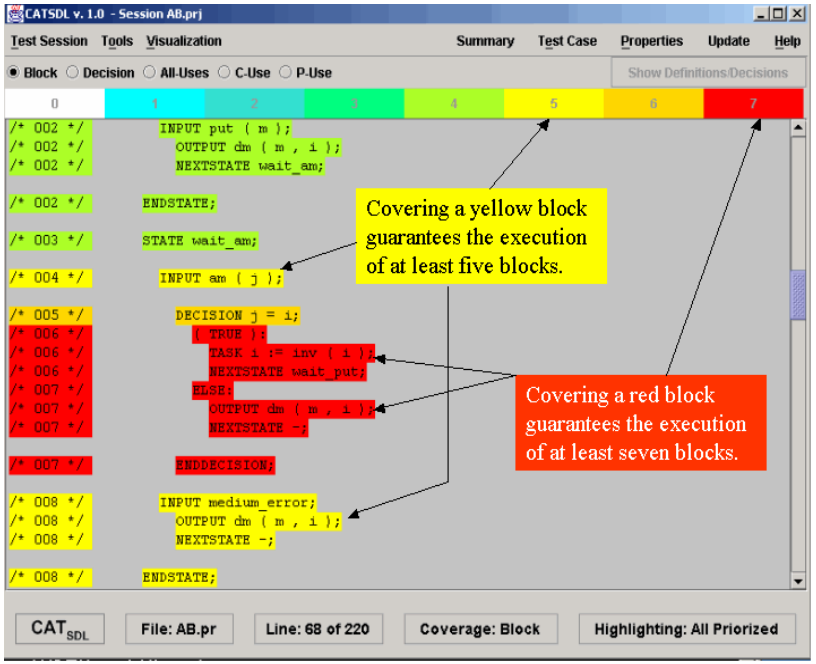


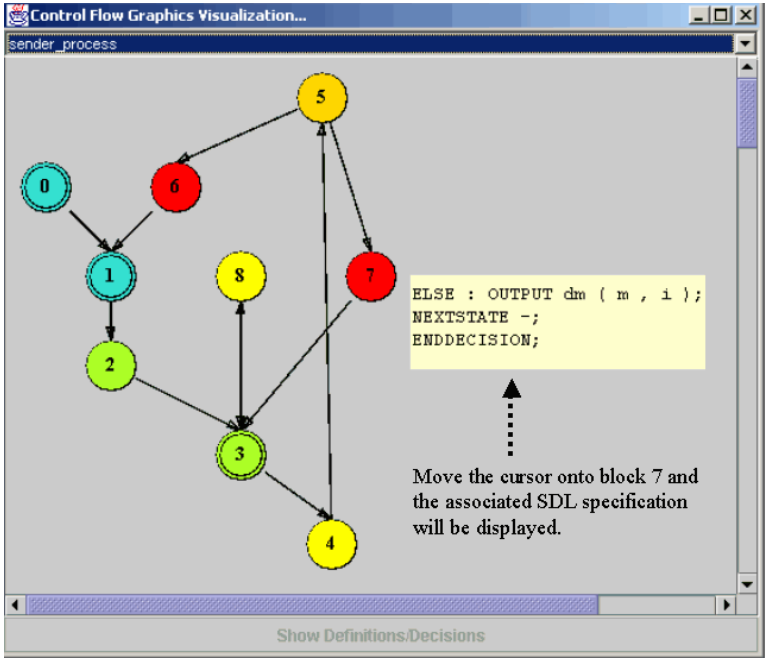
Analyzing the control-flow graph of an SDL specification to find the dominant blocks, decisions, etc.
For example, when a test case covers highly dominant blocks it will cover many other blocks.

- W. E. Wong *et al.*, "Coverage Testing Software Architectural Design in SDL," *Journal of Computer Networks*, 42(3):359-374, June 2003.

Coverage Testing SDL Specifications (2)

- Visualizing coverage in SDL specification and its control-flow graph

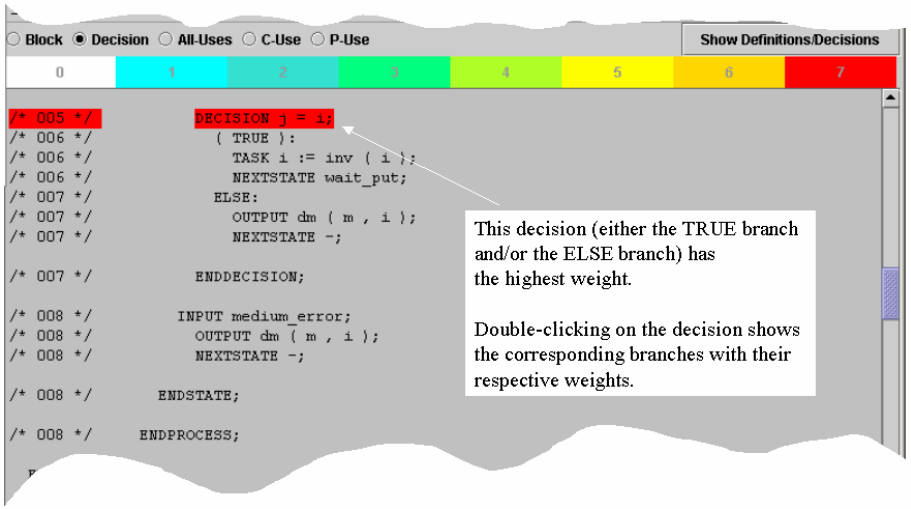


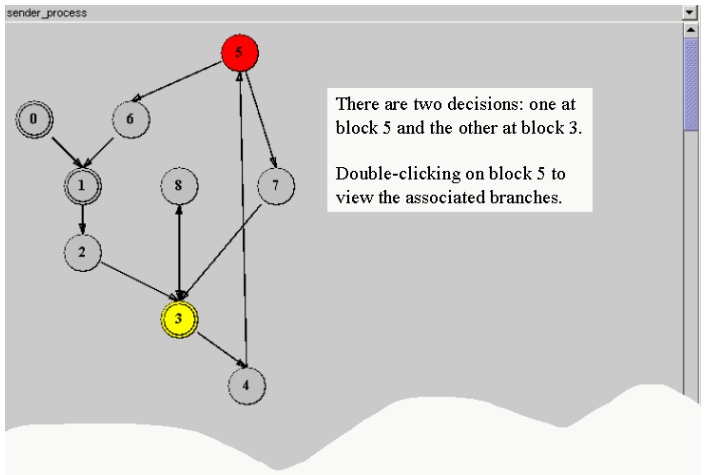


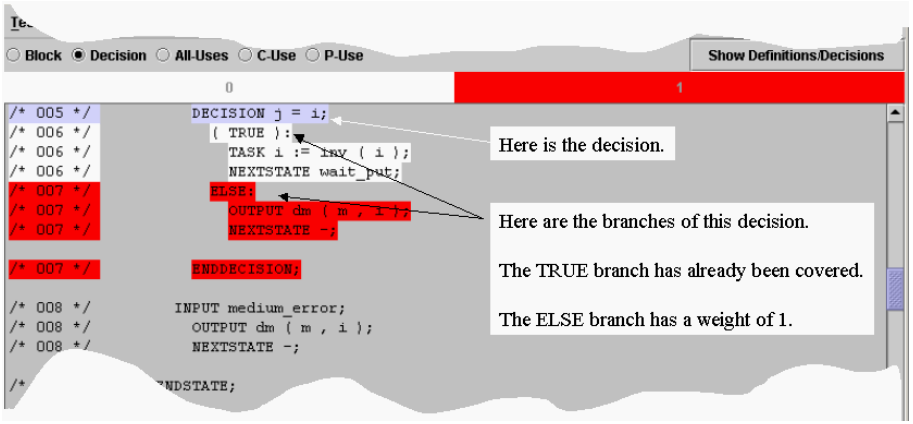
the textual representation — the corresponding control flow graph

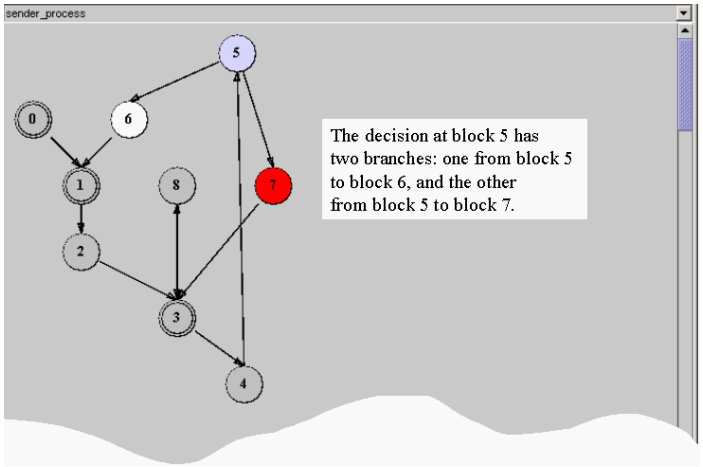
- A control flow graph is generated for each SDL process
- The textual representation displays the SDL source code, whereas the control flow graph makes its flow of control more evident

Coverage Testing SDL Specifications (3)







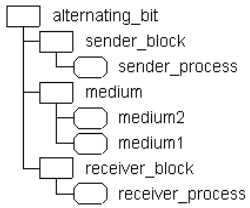


Design Philosophy for Testing/Maintenance Tools

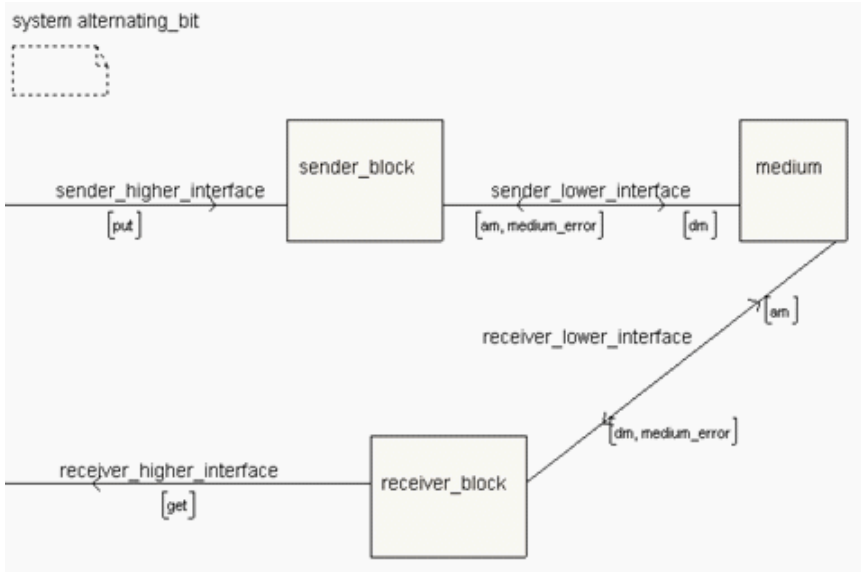
- When you develop a testing/maintenance tool, you should consider
  - Ease of use
  - Visualization
  - Prioritization
  - Granularity
  - Incrementability
  - Extensibility
  - Portability
  - etc.

Alternating Bit Protocol (1)

- The alternating bit protocol, which is a simple form of the "sliding window protocol" with a window size of 1, is used as the example.
- It can be used to provide *reliable communication* over *non-reliable network channels* through a *one-bit sequence number* (which alternates between 0 and 1)
  in each message.



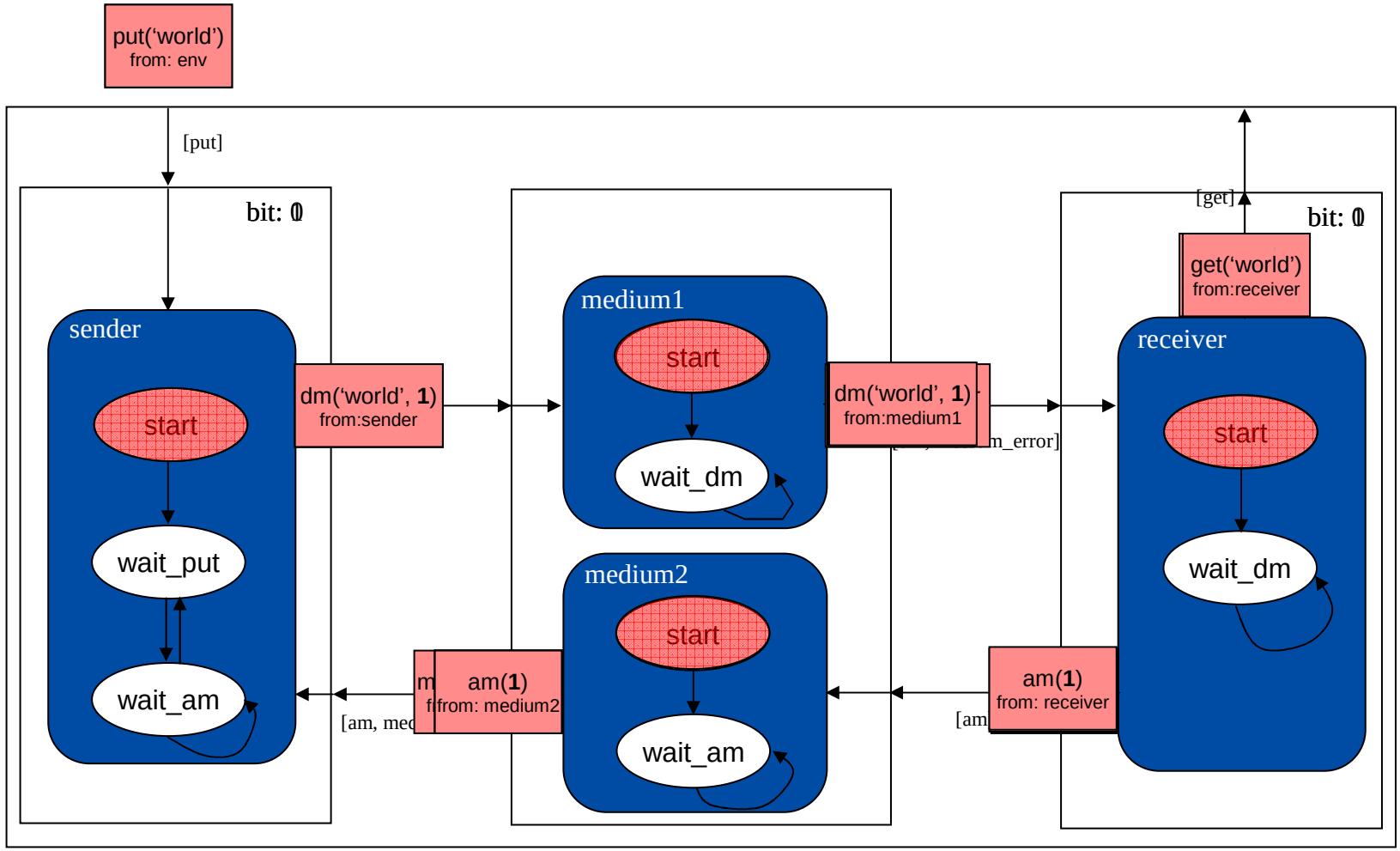
rw	alternating_bit.ssy
rw	sender_block.sbk
rw	sender_process.spr
rw	medium.sbk
rw	medium2.spr
rw	medium1.spr
rw	receiver_block.sbk
rw	receiver_process.spr



Alternating Bit Protocol (2)

- The alternating bit protocol is constituted by a sender and a receiver who exchange messages through two channels, Medium1 and Medium 2.
  - When the sender sends a message (containing a protocol bit, 0 or 1) to the receiver through Medium 1, it sends the message *repeatedly* (with the corresponding protocol bit) until receiving an acknowledgment from the receiver that contains the same protocol bit as the message being sent.
  - When the receiver receives a message, it sends an acknowledgment to the sender through Medium 2 and includes the protocol bit of the message received.
  - The first time the message is received, the protocol delivers the message for processing. *Subsequent messages with the same bit are simply acknowledged.*
  - When the sender receives an acknowledgment containing the same bit as the message it is currently transmitting, *it stops transmitting that message*, *flips the protocol bit*, and repeats the protocol for the next message.
  - *This implies that the sender associates each message with a protocol bit which is alternated between 0 and 1 to differentiate consecutive messages.*

Alternating Bit Protocol (3)



Next

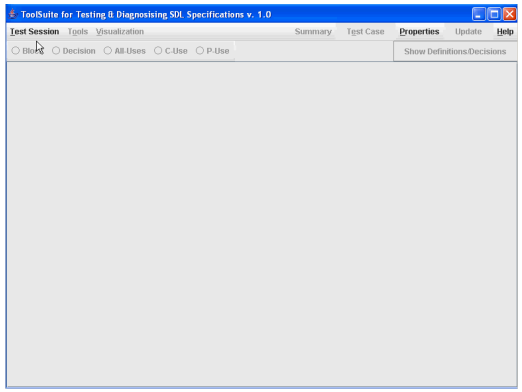
Questions

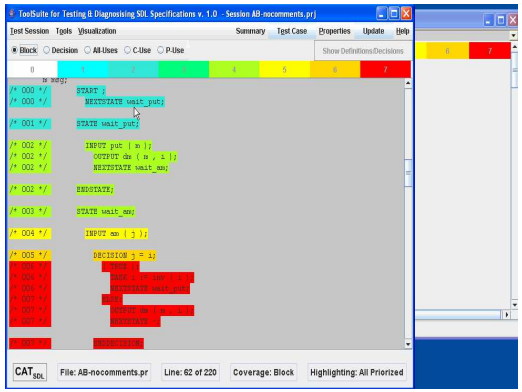
- Do you know how much of your SDL code has been tested?
- What is still missing?
- How to generate additional test cases to execute the uncovered code in an effective way?

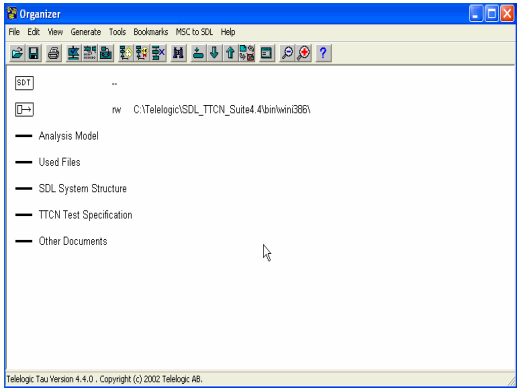
CAT_{SDL}: A Coverage Analysis Tool for SDL Specifications

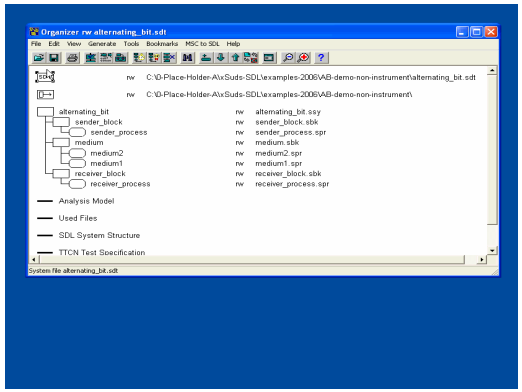
- Given an SDL specification, CAT_{SDL} performs instrumentation on it by inserting a probe, a user-defined function, at appropriate locations.
- The resulting instrumented specification is then exported to an SDL simulator, (Telelogic Tau in our case) for simulation.
- A file is created to record the trace information during the simulation.
- As subsequent simulation continues, the trace information is appended to the trace file which is then exported back to CAT_{SDL} for coverage analysis.

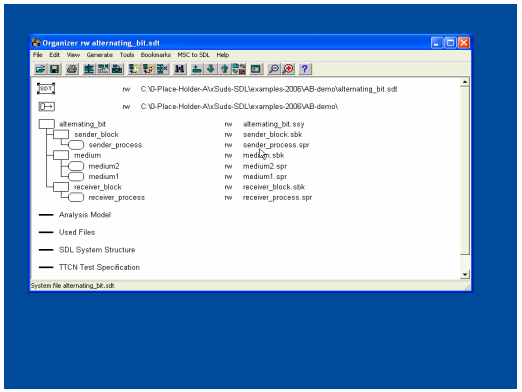
CAT_{SDL} Demo

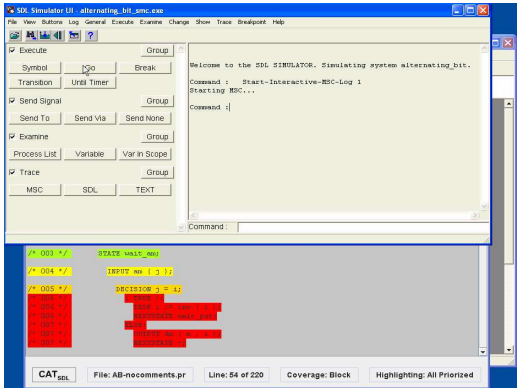




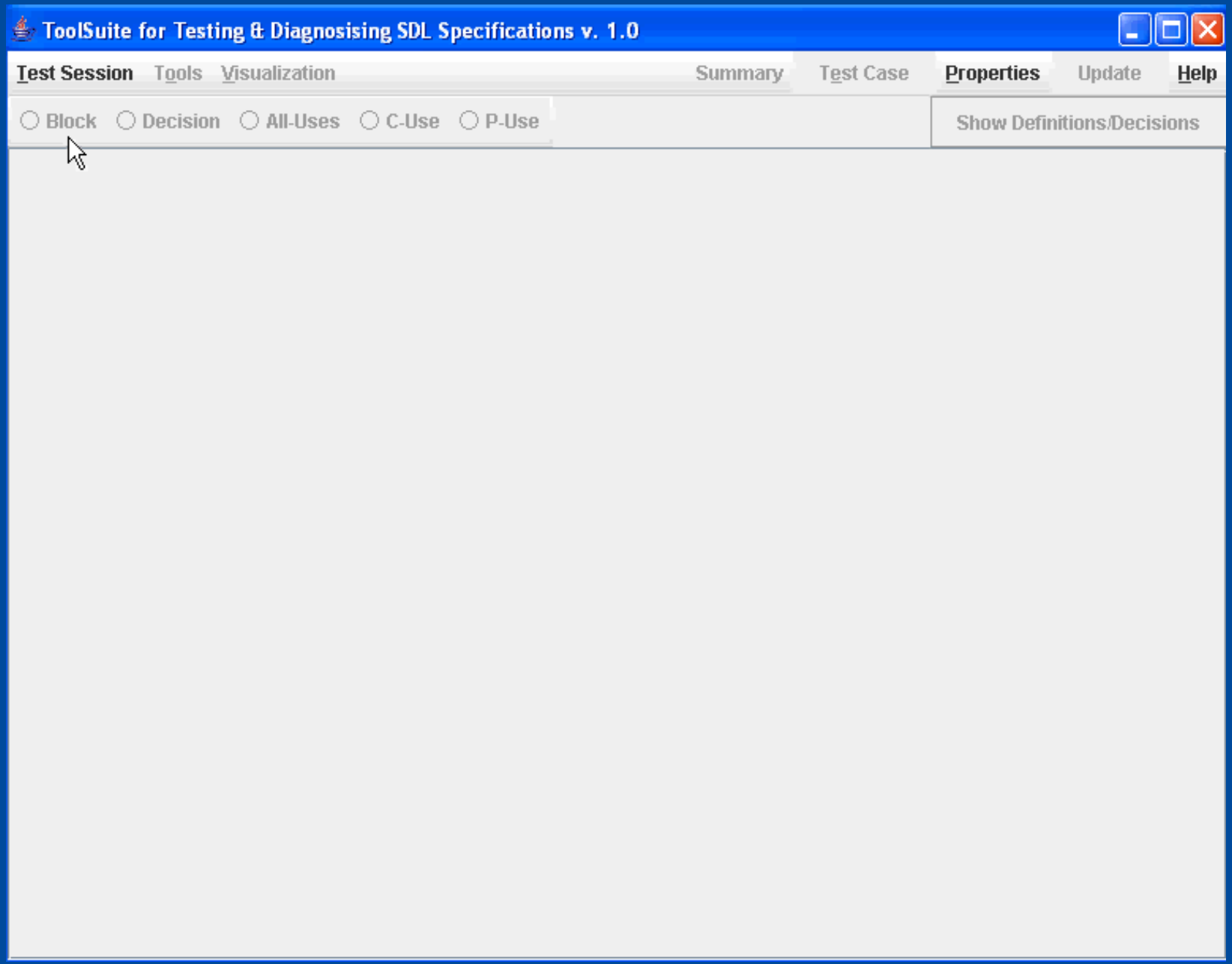




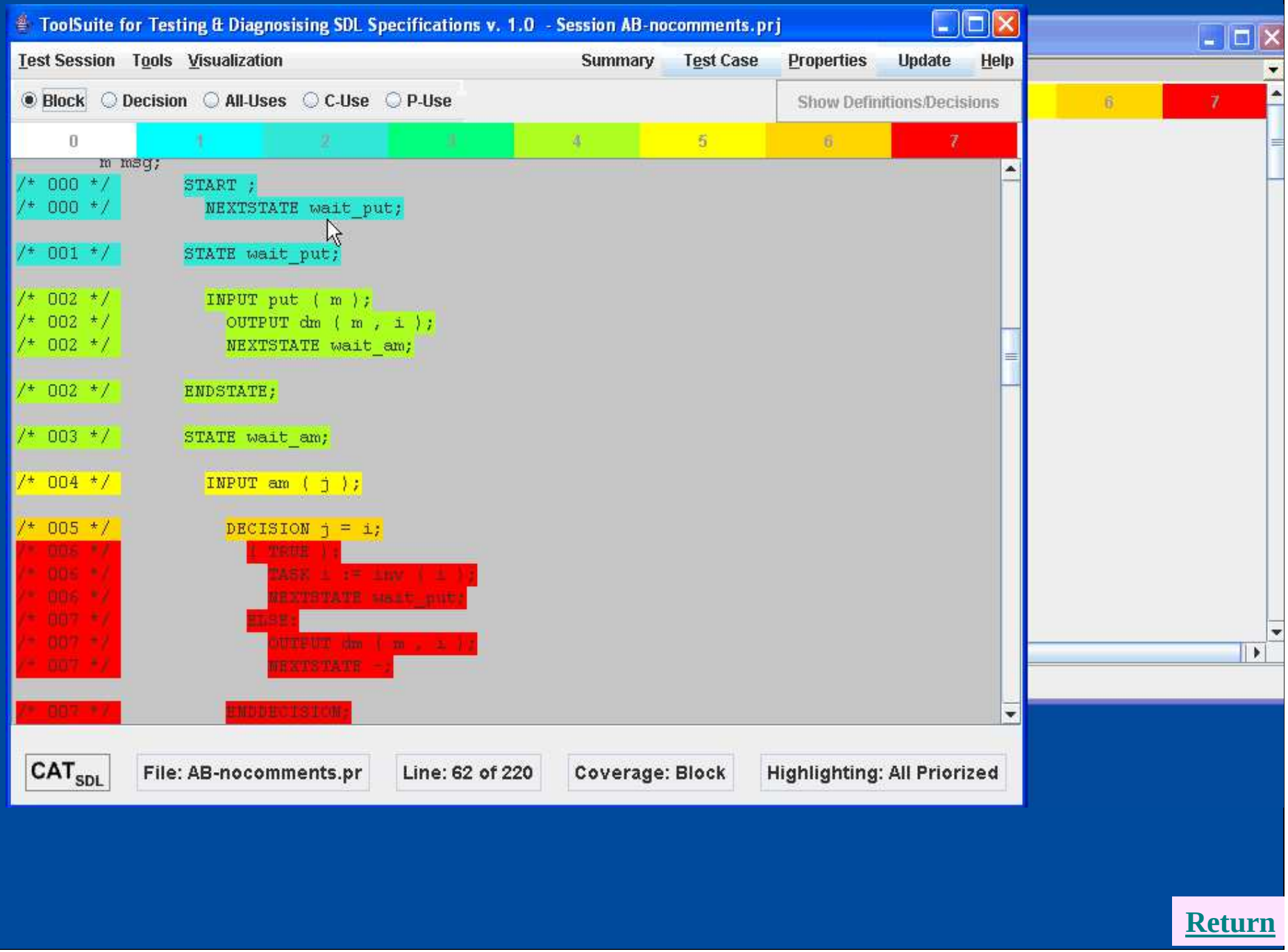




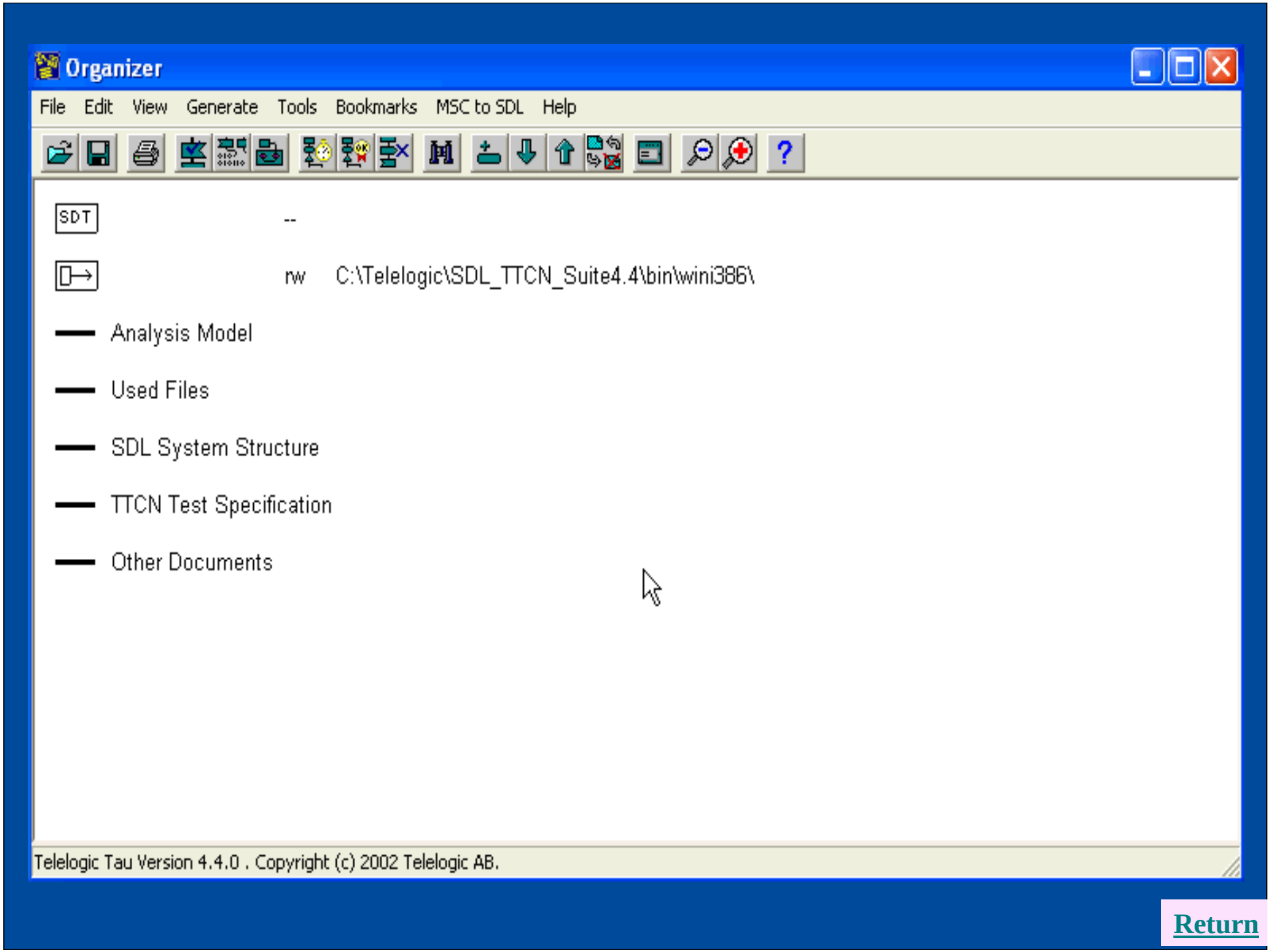
Previous Next


ToolSuite for Testing & Diagnosising SDL Specifications v. 1.0
Test Session
Tools
Visualization
Summary
Test Case
Properties
Update
Help
Block
Decision
All-Uses
C-Use
P-Use
Show Definitions/Decisions

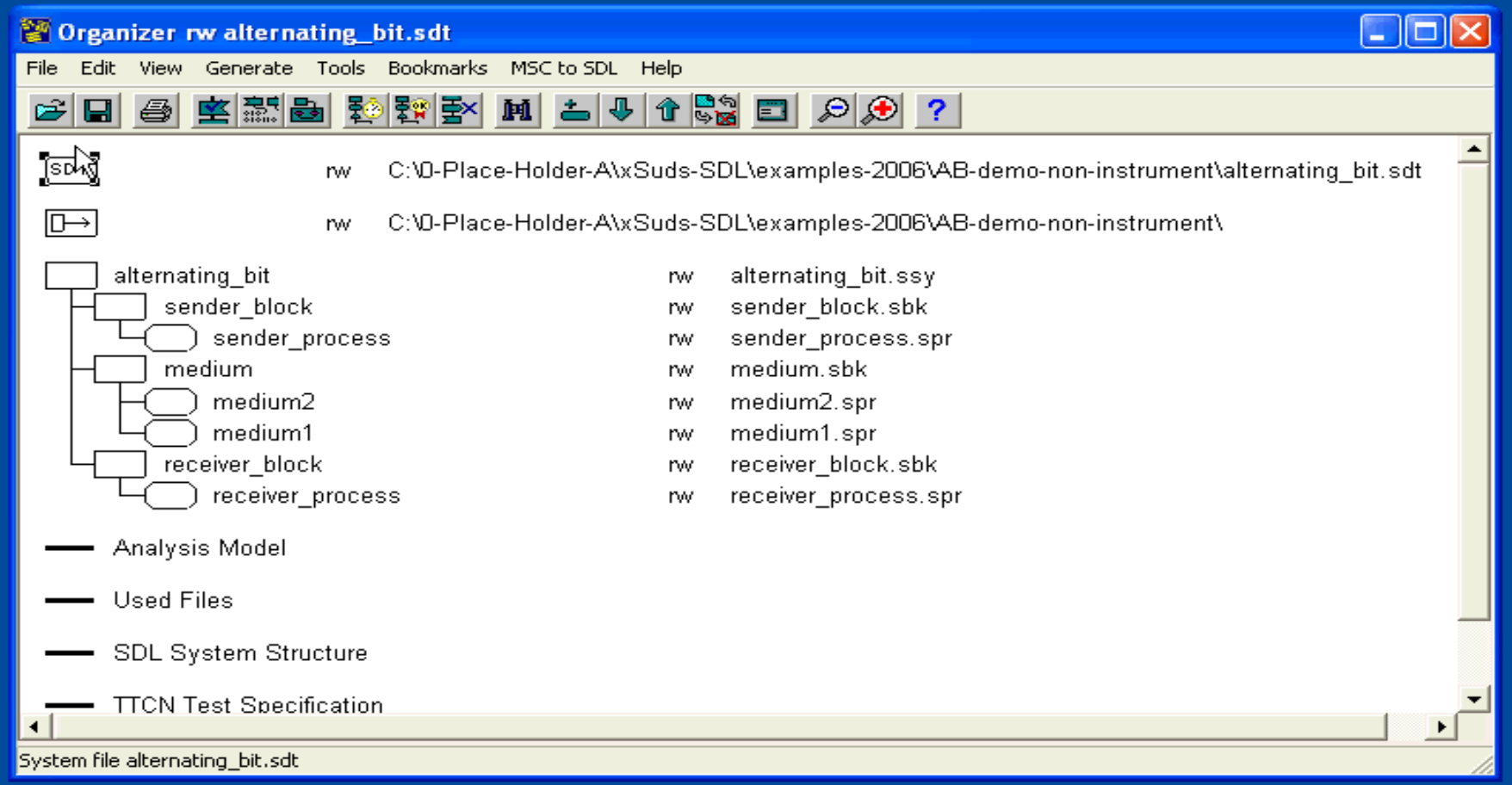
Return


ToolSuite for Testing & Diagnosising SDL Specifications v. 1.0 - Session AB-nocomments.prj
Test Session
Tools
Visualization
Summary
Test Case
Properties
Update
Help
Block
Decision
All-Uses
C-Use
P-Use
Show Definitions/Decisions
0
1
2
3
4
5
6
7
m msg;
/* 000 */ START ;
/* 000 */ NEXTSTATE wait_put;
/* 001 */ STATE wait_put;
/* 002 */ INPUT put (m);
/* 002 */ OUTPUT dm (m , i);
/* 002 */ NEXTSTATE wait_am;
/* 002 */ ENDSTATE;
/* 003 */ STATE wait_am;
/* 004 */ INPUT am (j);
/* 005 */ DECISION j = i;
/* 006 */ (TRUE):
/* 006 */ TASK i := inv (i);
/* 006 */ NEXTSTATE wait_put;
/* 007 */ ELSE:
/* 007 */ OUTPUT dm (m , i);
/* 007 */ NEXTSTATE -;
/* 007 */ ENDDECISION;
CAT SDL
File: AB-nocomments.pr
Line: 62 of 220
Coverage: Block
Highlighting: All Priorized

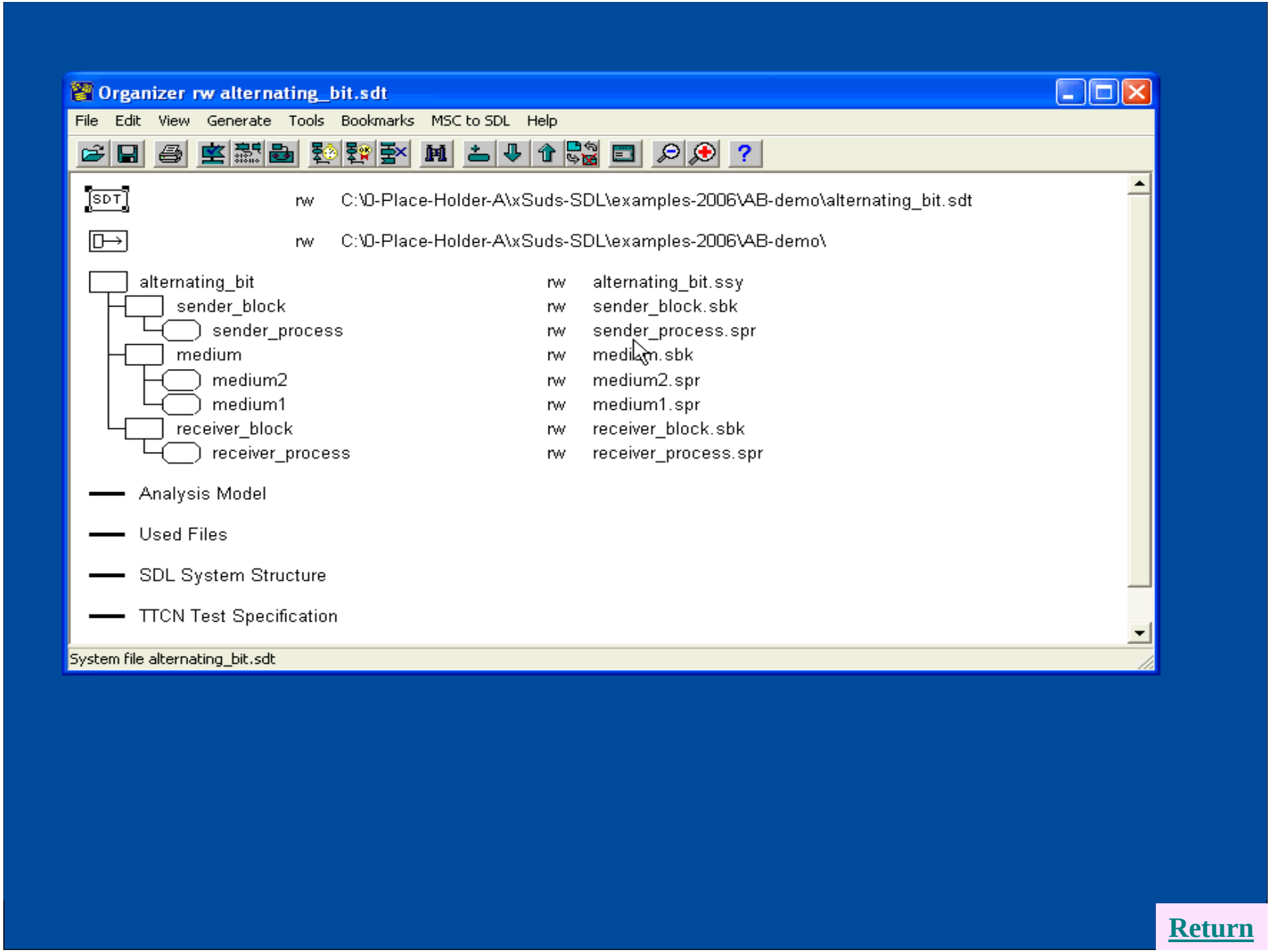
Return


Organizer
File Edit View Generate Tools Bookmarks MSC to SDL Help
SDT
--
rw C:\Telelogic\SDL_TTCN_Suite4.4\bin\wini386\
Analysis Model
Used Files
SDL System Structure
TTCN Test Specification
Other Documents
Telelogic Tau Version 4.4.0 . Copyright (c) 2002 Telelogic AB.

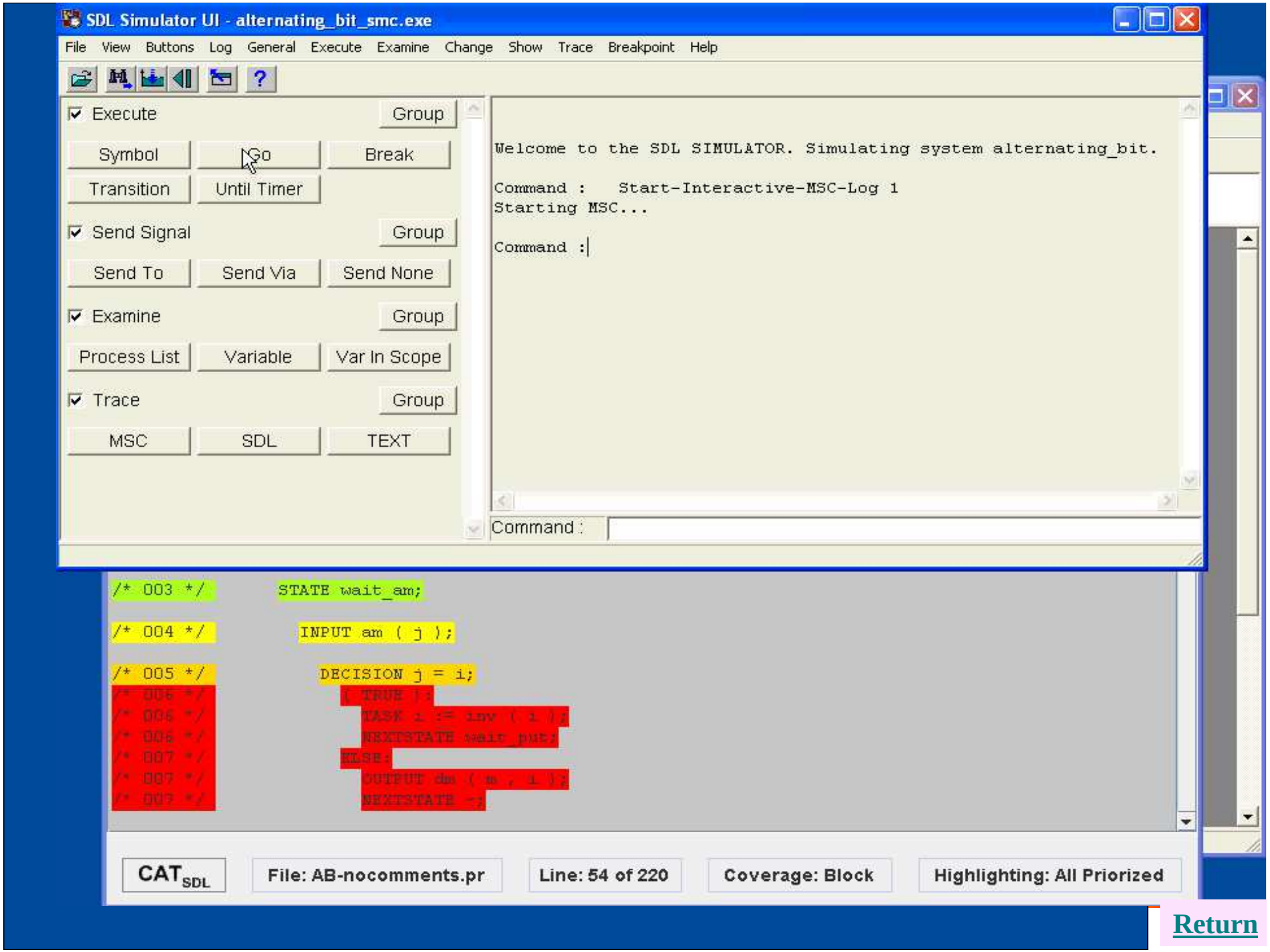
Return


Organizer rw alternating_bit.sdt
File Edit View Generate Tools Bookmarks MSC to SDL Help
rw C:\0-Place-Holder-A\xSuds-SDL\examples-2006\AB-demo-non-instrument\alternating_bit.sdt
rw C:\0-Place-Holder-A\xSuds-SDL\examples-2006\AB-demo-non-instrument\
alternating_bit rw alternating_bit.ssy
sender_block rw sender_block.sbk
sender_process rw sender_process.spr
medium rw medium.sbk
medium2 rw medium2.spr
medium1 rw medium1.spr
receiver_block rw receiver_block.sbk
receiver_process rw receiver_process.spr
Analysis Model
Used Files
SDL System Structure
TTCN Test Specification
System file alternating_bit.sdt

Return


Organizer rw alternating_bit.sdt
File Edit View Generate Tools Bookmarks MSC to SDL Help
SDT
rw C:\0-Place-Holder-A\xSuds-SDL\examples-2006\AB-demo\alternating_bit.sdt
rw C:\0-Place-Holder-A\xSuds-SDL\examples-2006\AB-demo\
alternating_bit rw alternating_bit.ssy
sender_block rw sender_block.sbk
sender_process rw sender_process.spr
medium rw medium.sbk
medium2 rw medium2.spr
medium1 rw medium1.spr
receiver_block rw receiver_block.sbk
receiver_process rw receiver_process.spr
Analysis Model
Used Files
SDL System Structure
TTCN Test Specification
System file alternating_bit.sdt

Return


SDL Simulator UI - alternating_bit_smc.exe
File View Buttons Log General Execute Examine Change Show Trace Breakpoint Help
Execute
Group
Symbol
Go
Break
Transition
Until Timer
Send Signal
Group
Send To
Send Via
Send None
Examine
Group
Process List
Variable
Var In Scope
Trace
Group
MSC
SDL
TEXT
Welcome to the SDL SIMULATOR. Simulating system alternating_bit.
Command : Start-Interactive-MSC-Log 1
Starting MSC...
Command :
Command :
/* 003 */ STATE wait_am;
/* 004 */ INPUT am (j);
/* 005 */ DECISION j = i;
CAT SDL
File: AB-nocomments.pr
Line: 54 of 220
Coverage: Block
Highlighting: All Priorized

Return

Research Issues on Coverage Testing (1)

- *How to reduce the amount of instrumentation?*
  - That is, how to *reduce the number of probes inserted in the source code?*
    - One probe per line of code
    - One probe per block
    - One probe per superblock
    - What are the pros and cons?
- *What kind of information should be collected at run time?*
  - Executed or not executed
  - Execution counts of each testable attribute
  - Execution sequence
- *What kind of information should be collected at parsing?*
  - Number of testable attributes
  - Locations of each testable attribute
- Can the techniques used for coverage testing software on Windows be applied to coverage testing embedded software? Real-time applications?

Research Issues on Coverage Testing (2)

- In addition to reporting what has been tested and what is still missing, we would also like to
  - *Decompose coverage information based on users' needs* (do not just report a single number with respect to all the tests for the entire program)
  - *Provide useful hints for efficient test generation to effectively increase the coverage of the program being tested*
  - *Use coverage as a filter to show how the number of test cases* can be reduced significantly without sacrificing the overall code coverage
  - Test cases in the reduced subset have a higher priority to be executed when revalidating the program during the regression testing
  - *Conduct effective fault localization* based on how the program is executed by each test