Diagnostics and Testability

Eric Gould
DSI International
October 2002

Diagnostics

- A process that correlates the results of multiple tests to determine overall system status and generate hypotheses (fault groups) for maintenance / remediation.
  - Testing vs. Diagnostics
  - Determines overall system status
  - Generates hypotheses

Testability

- A design characteristic which allows the status (operable, inoperable, degraded) of an item to be determined and the isolation of faults within the item to be performed in a timely manner
  - Characteristic of a design
  - Enables determination of item status
  - Facilitates testing / diagnostics

The Two "Testabilities"

- Design for Test
  - Good design practices that facilitate Testing
  - Usually performed by designers
- Design for Diagnosis ("Diagnosability")
  - Optimization of design to facilitate Diagnostics (e.g. Test Point Placement)
  - Optimization of diagnostic strategies
  - Usually performed by designers or by analysts in conjunction with designers

Diagnostic Engineering

- The engineering discipline through which the diagnostic capability of a system or device is developed assessed and optimized. Diagnostic Engineering is comprised of three inter-related processes:
  - **Diagnostic Development** (test strategy generation)
  - **Diagnostic Assessment** (evaluates both diagnostics & design)
  - **Design Development** (improvements to facilitate diagnosis)

The Diagnostic Engineering Process

Diagnostic Development

- Diagnostics Developed Simultaneously with Design
- Updated based on Iterative Assessments

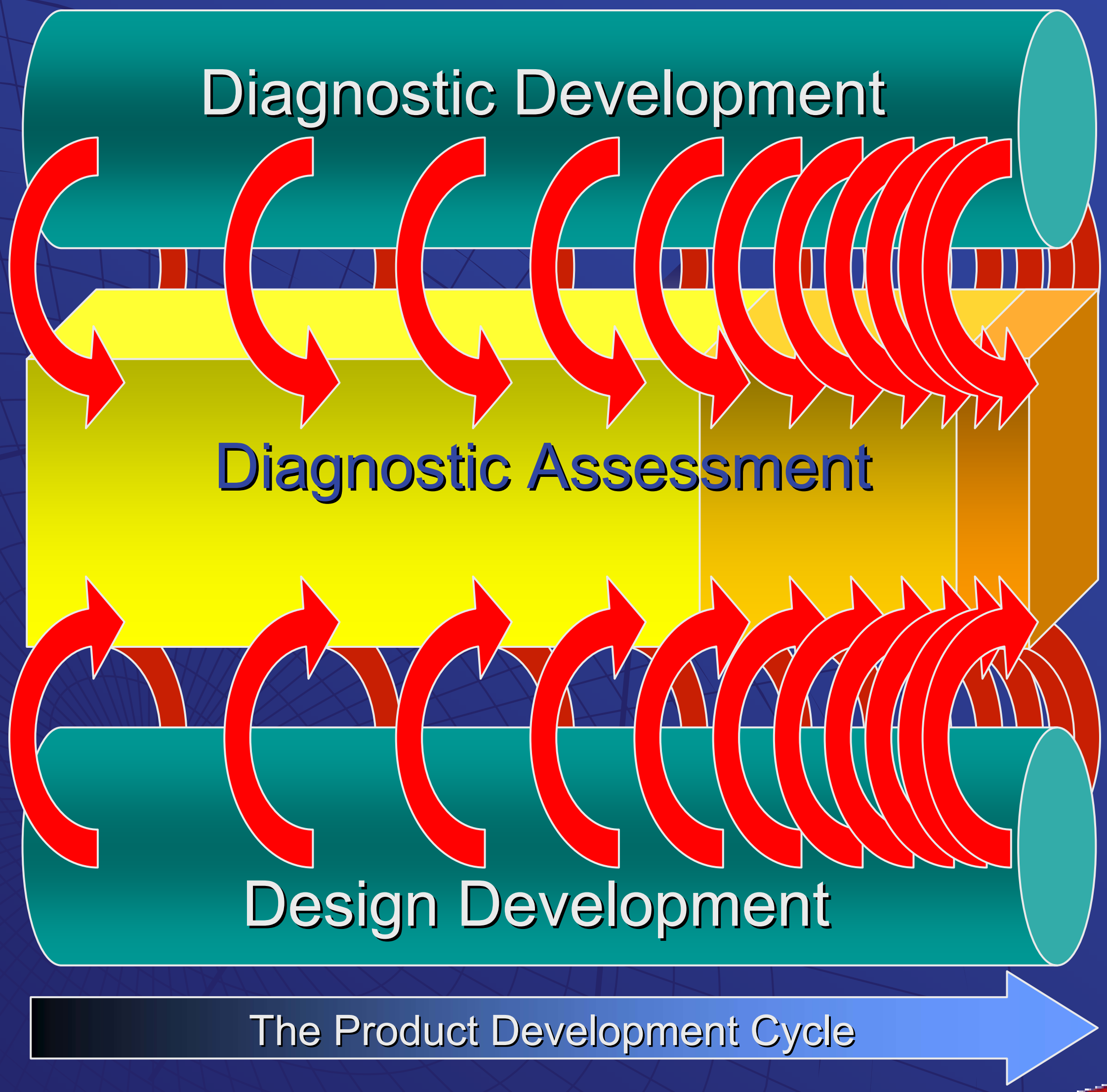
Diagnostic Assessment

- Evaluates Diagnostics Together with Design
- Provides Feedback to Both Diagnostics and Design
- Used to Determine Requirement Allocations
- Assessments Become More Frequent As Design and Diagnostics Mature

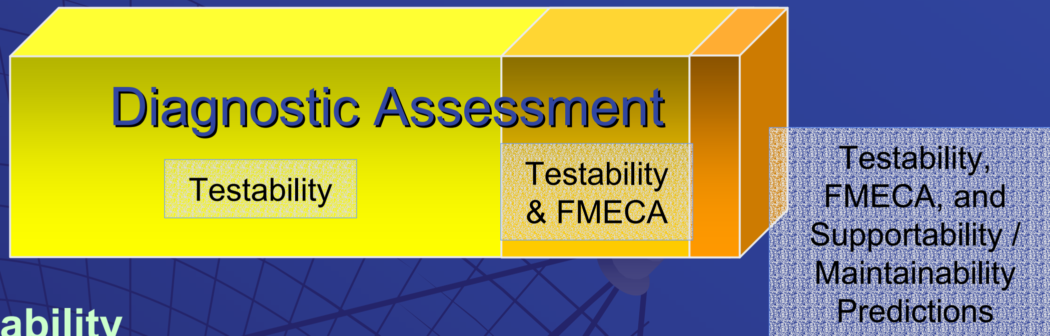
Design Development

- Diagnosability Assessed in Earliest Development Phases
- Updated based on Iterative Assessments

The Diagnostic Engineering Development Cycle



Phases of Assessment in the Diagnostic Engineering Development Cycle



Testability

- Can commence in the earliest design phases (Should *not* be postponed until after FMECA)
- Metrics are meaningful while the design is still in flux
- Provides useful feedback throughout Diagnostic Engineering Cycle

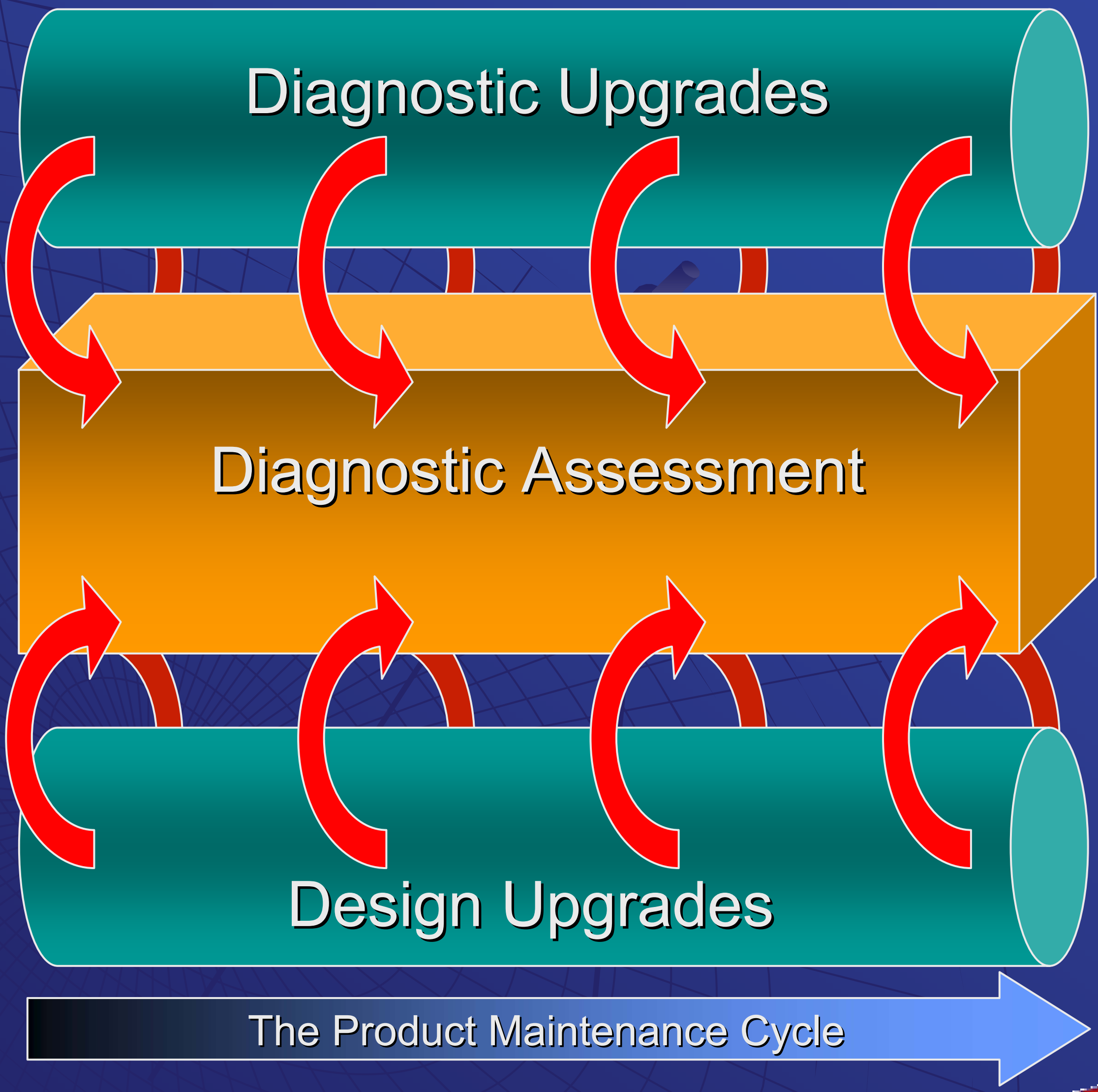
Failure Modes, Effects & Criticality Analysis (FMECA)

- Requires that specific component failure modes be identified
- Typically not performed until design is well established
- Effort reduced if based on same data models as Testability

Supportability & Maintainability Predictions

- Requires design and diagnostics to be well established
- Results in changes to diagnostics / maintenance plans more often than in modifications to the design

The Diagnostic Engineering Maintenance Cycle



Two Points That Are Important Enough To Repeat *Ad Nauseum*

- Testability Analysis should be performed iteratively throughout the development and deployment of the design
- Testability Analysis should be performed starting in the earliest development phase in which feedback on design diagnosability may be useful

Some Characteristics of *eXpress*

How *eXpress* facilitates iterative analysis:

- Test definitions are automatically updated as model matures
- Robust attribute engine allows *eXpress* to act as a data governing tool
- Open (COM) interface allows *eXpress* to be easily integrated into any process

How *eXpress* facilitates early analysis:

- *eXpress* doesn't simply allow top-down analysis, it encourages it
- Functional dependencies can be analyzed before failure modes are known
- *eXpress* analysis produces metrics that are useful even when minimal design details are available

Dependency Models

- Dependency Models are representations of the behavior of a device or system in terms of the causal relationships between its different elements.

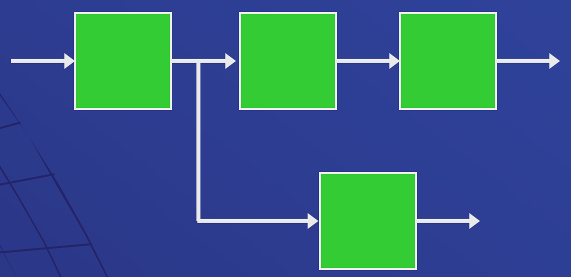
Event

Cause Event #

Dependency ^-o-#

^-o-o-#

Agent ^--------o-#



Two Types of Dependency Models

- *Functional* Dependency Models represent how a device or system behaves when operating properly.

```
#            Function
                      Nominal Event
^-o-#
```

- *Failure Mode* Dependency Models represent the different ways in which a device or system can malfunction.

```
#            Failure Mode
                      Non-nominal Event
^-o-#
```

Functional and Failure Mode Dependency Models

- *Functional* Dependency Models
  - May be hardware-independent
  - Can be developed early in the design process
  - Appropriate for representing component or system-level behavior
  - Fully describes design functionality
- *Failure Mode* Dependency Models
  - Must be hardware-dependent
  - Cannot be developed until relatively late in the design process
  - Typically used to represent component-level behavior
  - Often constrained to a fault universe

Hybrid Dependency Models in *eXpress*

- Hybrid Dependency Models represent the behavior of a device or system in terms of both functional and failure mode causes.
  - A functional model is first developed
  - Failure modes are overlaid over the functional model
  - Affected functions are identified for each failure mode
- Hybrid Dependency Models allow diagnostics to test in terms of either functions or failure modes.
  - Function and failure mode statuses are correlated during diagnostics
- Hybrid Dependency Models allow system diagnostic and FMECA analysis to be derived from the same database.

Diagnostic Dependency Models

- Diagnostic Dependency Models represent the different ways in which a device or system can be tested.

```
#
^-o-#
    ^-o-o-#
    ^--------o-#
```

Tests

Traditional Dependency Modeling

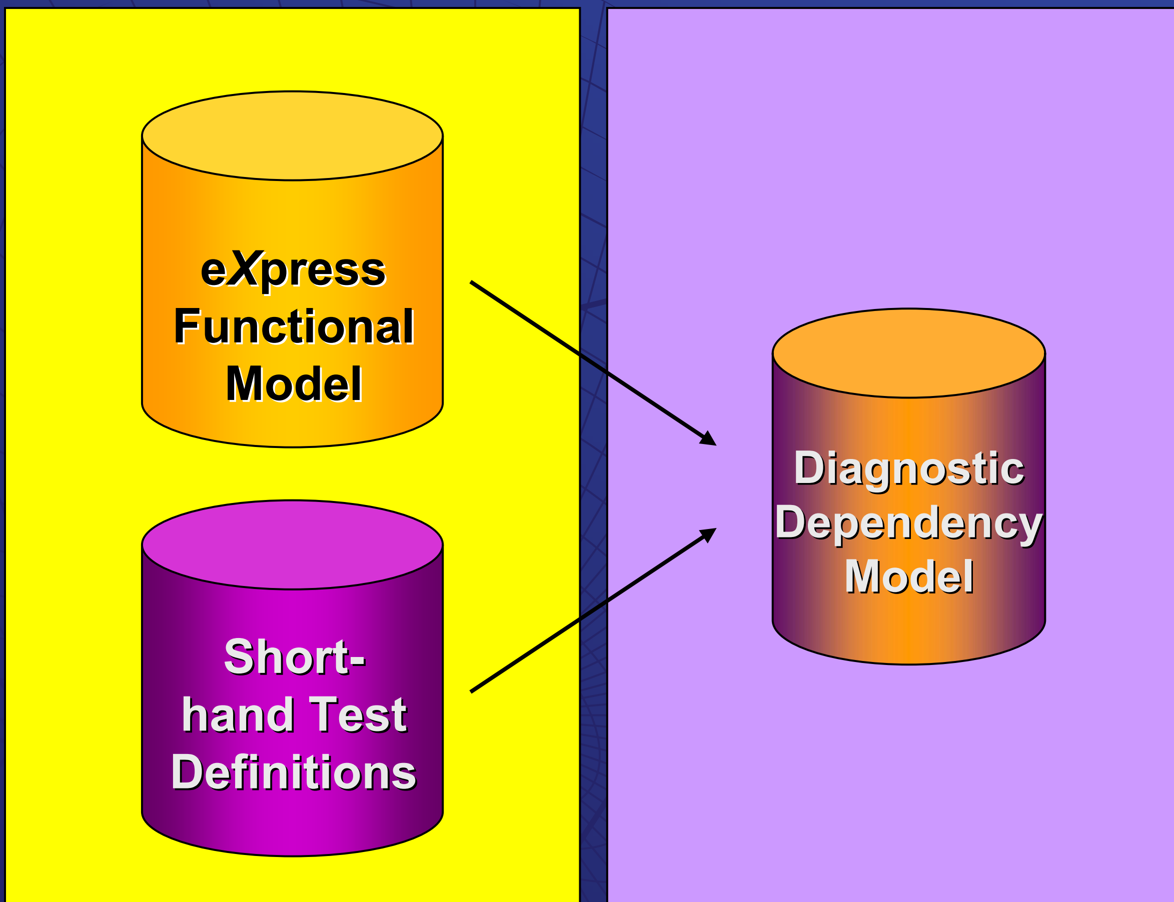
- First-Order Dependency Statements describe the elements of the design that have an immediate effect upon the results of the specified test(s).

```
#
^-o-#
    ^-o-o-#
    ^--------o-#
```

- Nth-Order Dependency Statements describe all elements of the design that can impact the results of the specified tests. Nth-Order dependencies can either be derived from first-order dependencies, or entered by hand.

```
#
^-o-#
    ^-o-o-#
    ^--------o-#
```

eXpress Modeling

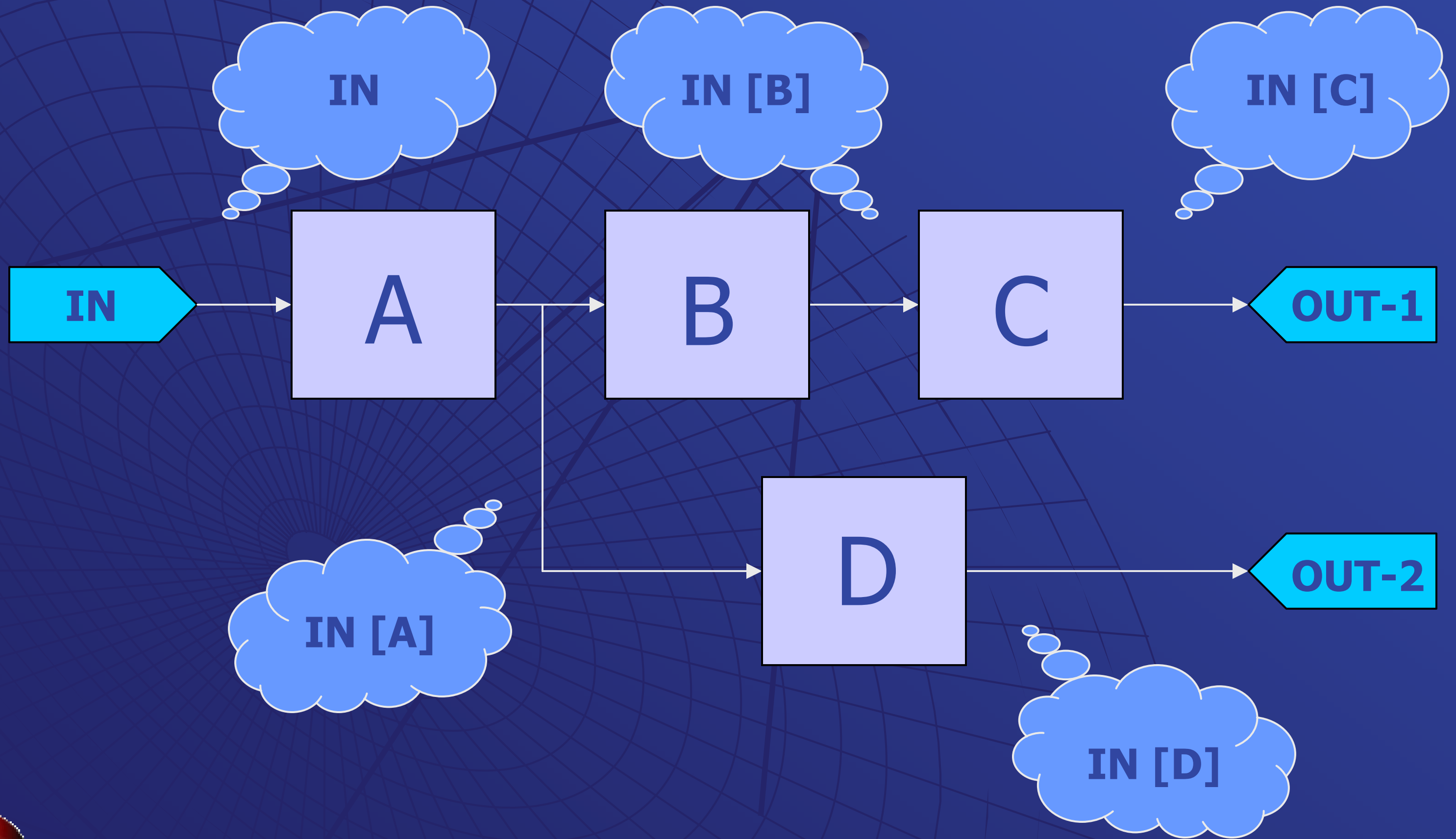


Topology, Functional Dependencies and Failure Modes are defined in an *eXpress* Model.

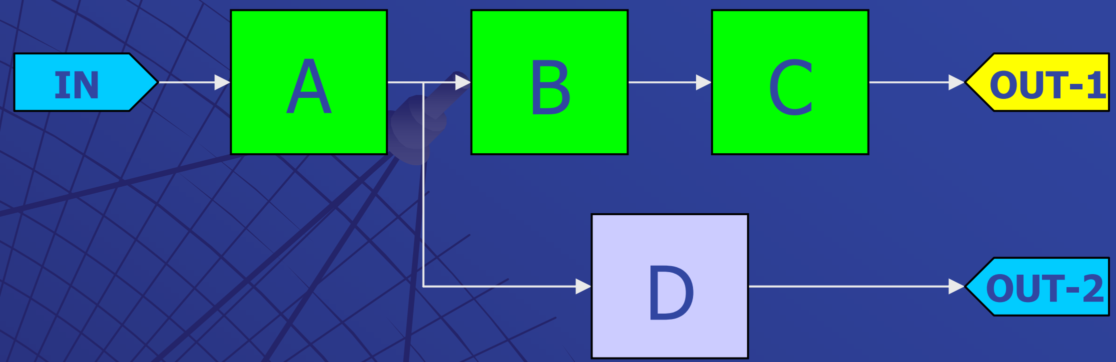
Next, Tests are added to the *eXpress* model using short-hand definitions.

eXpress automatically creates a full-ordered Diagnostic Dependency Model by overlaying the Test Definitions over the *eXpress* Model.

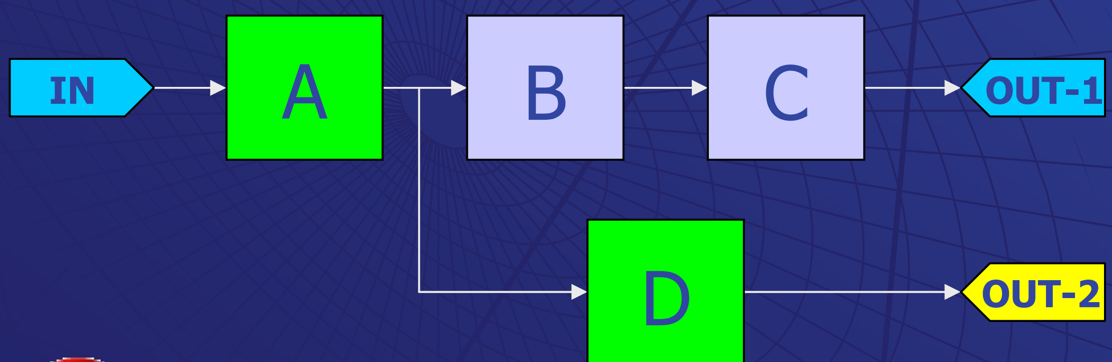
Functions Are Propagated As You Draw



Defining Tests at Outputs: Operational and User-Initiated Tests

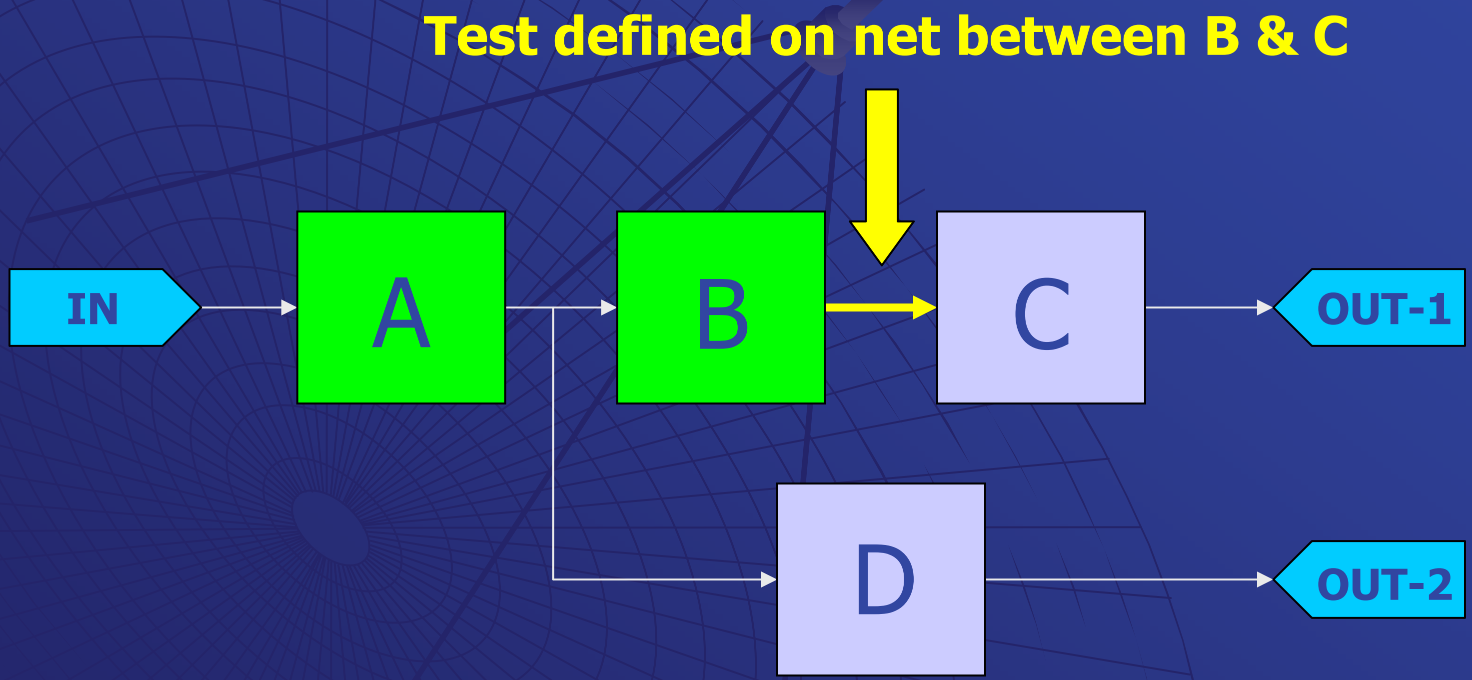


Coverage of Test Defined at OUT-1

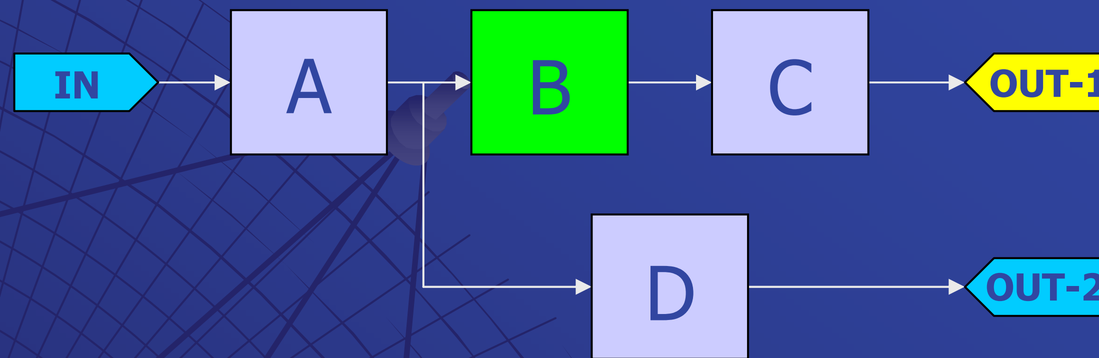


Coverage of Test Defined at OUT-2

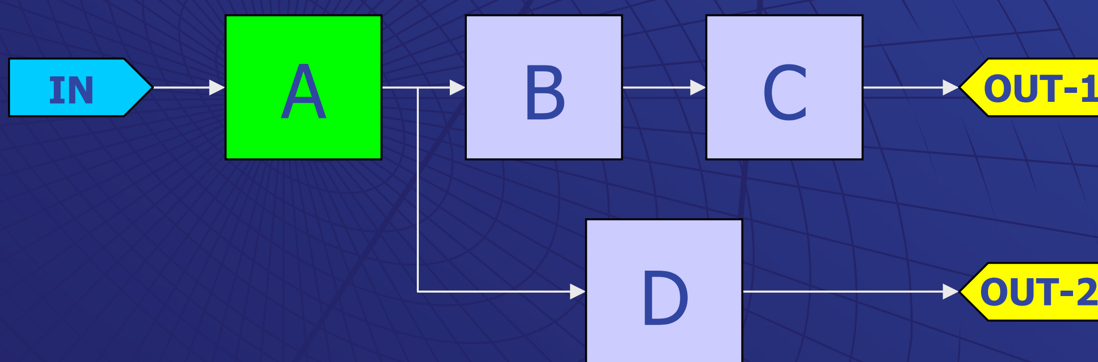
Defining Tests on Nets: Probe Tests



Defining Tests by Selecting Coverage: Signature Tests

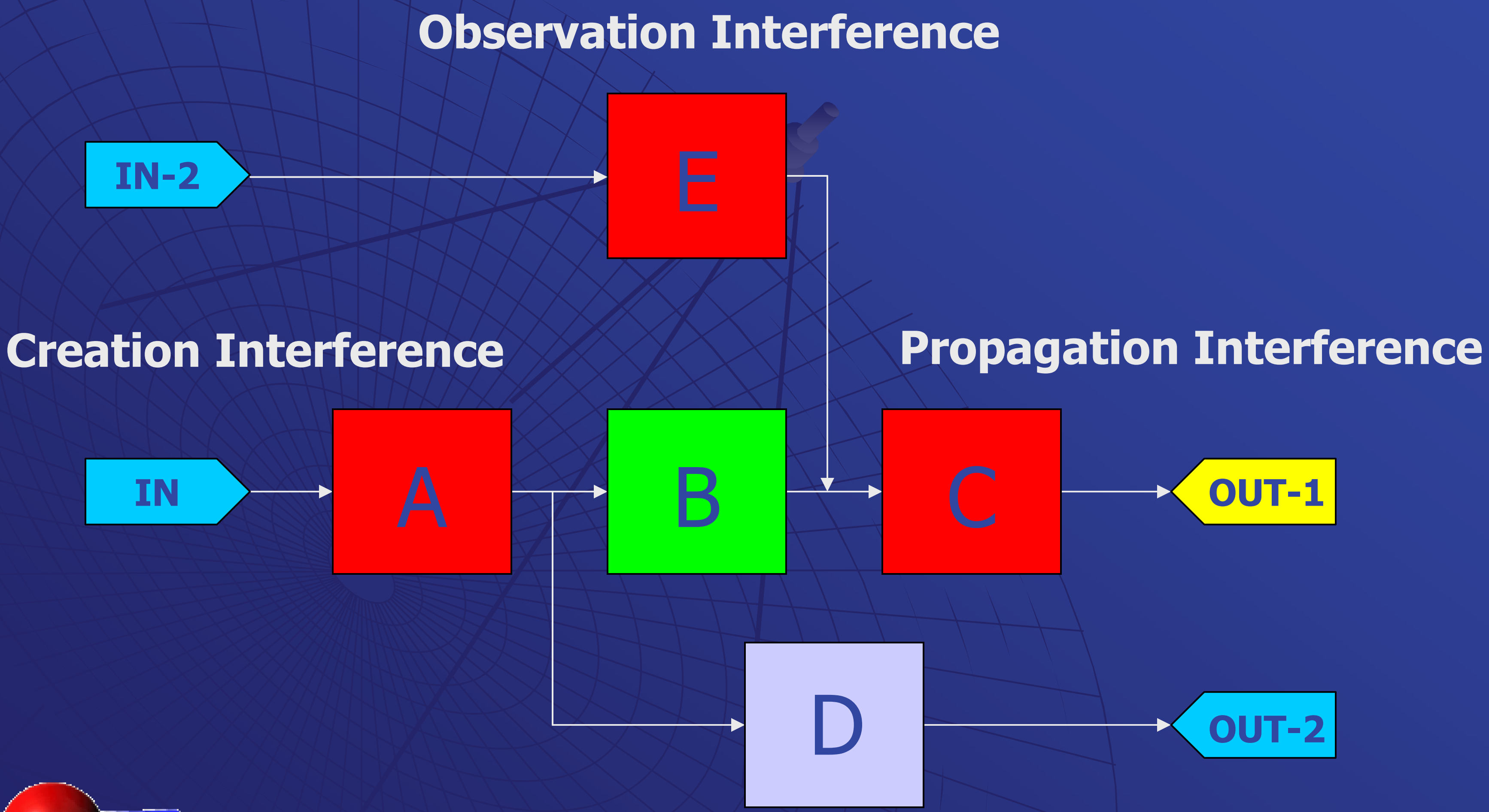


B is only visible at OUT-1



A is potentially testable at both OUT-1 and OUT-2

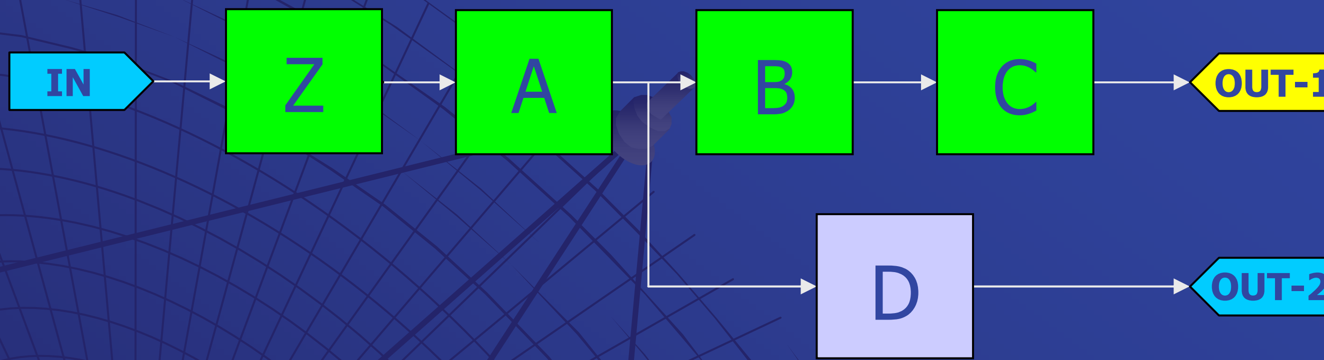
Signature Interference



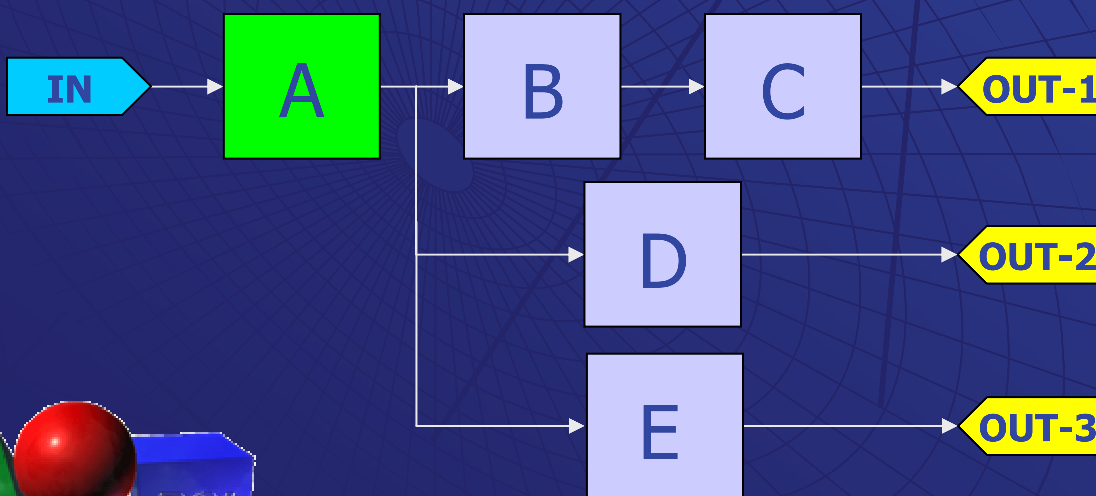
Test Asymmetry

- Results when the functions or failure modes that are exonerated (proved) when a test passes are not the same as those that are called into suspicion (detected) when that same test fails.
- Because when asymmetric tests are used for detection the portion of the design that is verified is not necessarily the same as that which is searched for malfunctions, there is a bifurcation of Fault Detection metrics: Faults Proven vs. Faults Detected

As Topology & Dependencies Change, Test Coverage is Automatically Updated



Z is automatically added to coverage



OUT-3 is a new candidate Test Location

Beyond Topology: Inspection Tests

Inspection tests should be used when the status of the part(s) can be determined…

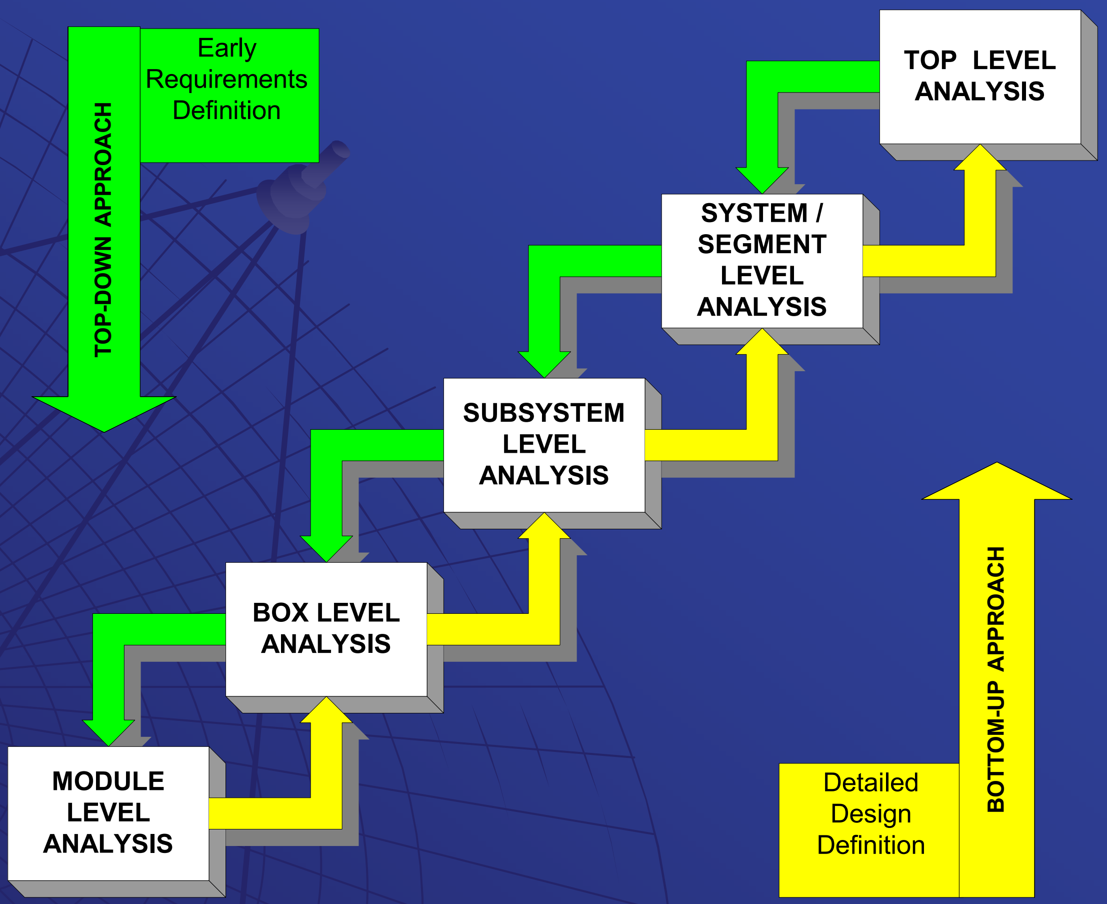
- Independent of the part(s)'s role in the system (visual inspection, external test equipment, etc.)
- Using "ambient" means (air temperature, sound, etc.)
- Using non-topological "rules"
- Using prognostic algorithms

Hierarchical Models in eXpress

Systems are typically modeled using a *meet-in-the-middle* approach.

Top-down models are used early in the design process to determine requirements allocations.

As design details become available, lower-level models are incorporated into the system from the *bottom up*.



Benefits of Hierarchical Modeling in *eXpress*

Top Down Modeling

- Enables Requirements Allocation Case Studies
- Facilitates Communication with Customer / Engineers

Bottom Up Modeling

- Provides Rollup of Design and Attribute Data
- Establishes Maintenance Levels for Diagnostics

"Meet-in-the-Middle" Modeling

- Ensures a Rigorous Approach to System Integration
- Allows Low-Level Assessments to be Evaluated in Context

The System Integrator Plays a Crucial Role in Development of the System Model

Upper-level Requirements

Upper-level Interface

Early Design Expectations

Contributions of Provider A

System Integrator

Lower-level Details

Lower-level Interface

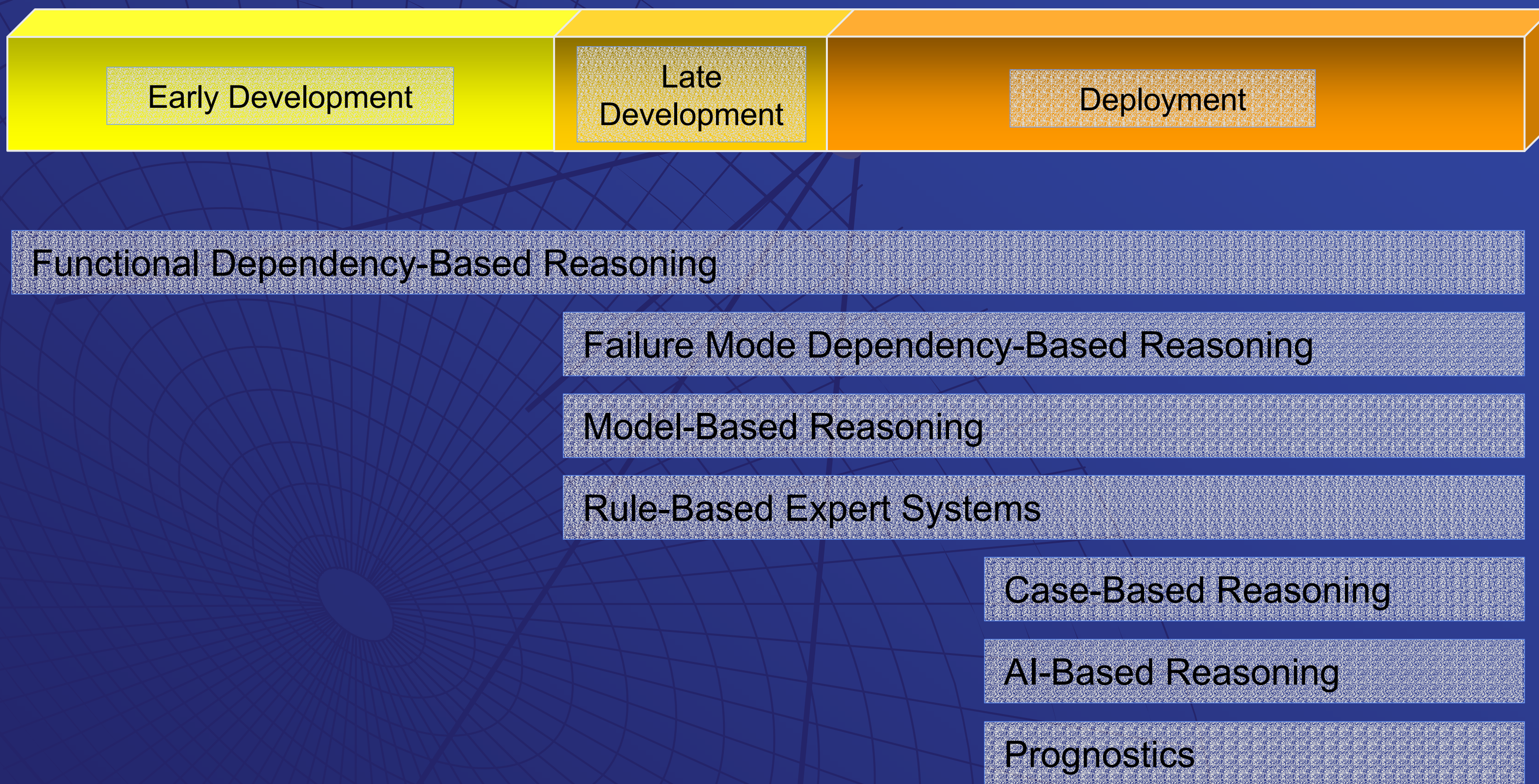
Later Design Realizations

Contributions of Provider B

Other Types of Diagnostic Models

- Rule-based: Expert Systems
- Case-based: Empirical Expert Systems
- MBR-based: Model-based Reasoners
- AI-based: Bayesian, Neural Net, etc.

Periods of Effectiveness for Different Types of Diagnostics



Advantages of *eXpress* Modeling

- A topological model can be developed before functions, failure modes and tests are introduced. This model can often be imported from engineering databases and easily compared against design schematics.

- Because failure modes are integrated with a full functional model, diagnostic predictions can be possible even when complete failure information is not available.

- Test definitions do not require extensive low-level updates every time that the design is modified. Instead, test definitions are used to automatically repopulate test coverage.

Benefits of *eXpress* Modeling

- *eXpress* models, because they can resemble design schematics, facilitate communication between engineers and analysts in different disciplines.
- *eXpress* models combine the strengths of both Functional and Failure Mode dependency models.
- *eXpress* models require fewer extensive, low-level revisions as a design matures, thereby lending themselves to iterative analyses in all phases of development.
- *eXpress* models can be profitably utilized in early phases of development – when diagnostic feedback can be most effectively used to improve the design.