Proposal Discussions
Ver .3

Ajay Khoche
Ajay.khoche@verigy.com
May, 2007

Agenda

- Scope Overview - Ajay
- Information flow model - Ajay
- Data Model review - Ajay
- Data Model Feedback and Discussions - All
  - Environment
  - Format specification
  - Validation & synchronization
  - Fail data collected (Buffer full condition)
- Additional Topics (Time permitting) - All

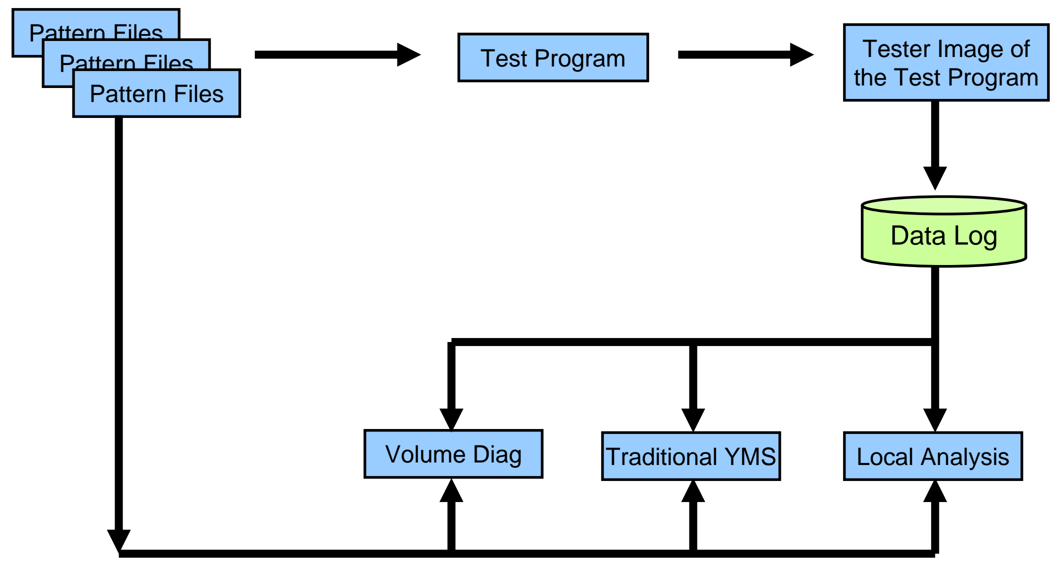
Scope

- Scan Fail datalog for volume diagnosis
- fail logging at Wafer as well as package test
- Format should allow capture of millions of failures (what one would expect in 2010)
- Both cycle and pattern based datalog support
- STDF V4 as the format (Using DGRs)

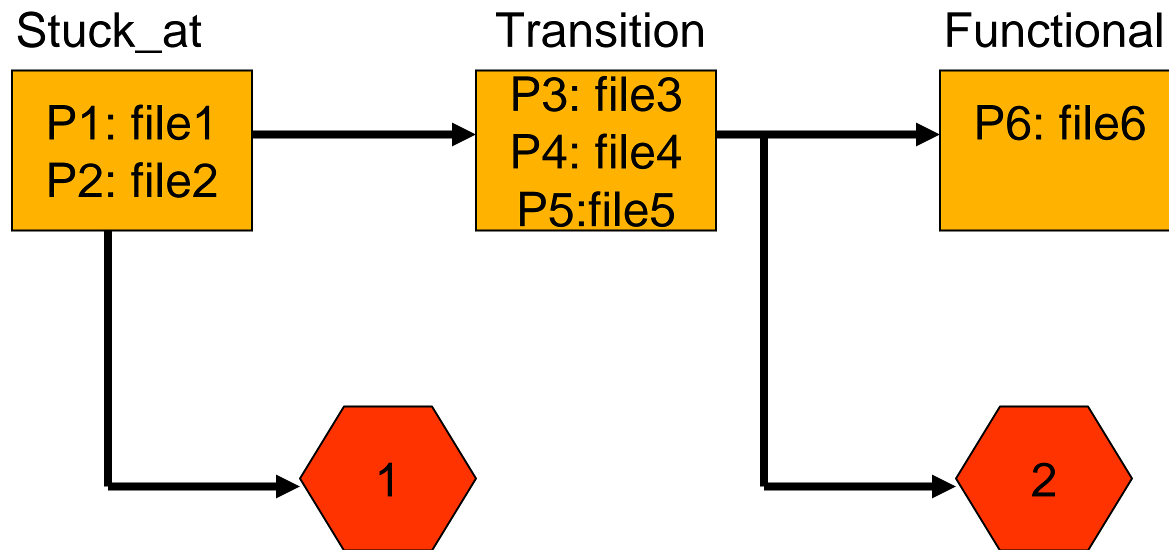
Attributes of the Standard Format

- Data Sufficiency: Meet the needs of downstream tools
- Data Efficiency : No/Low throughput impact and low data volume
- Data Integrity: provide consistency checking
- Data Integration: Must integrate well with other datalog formats

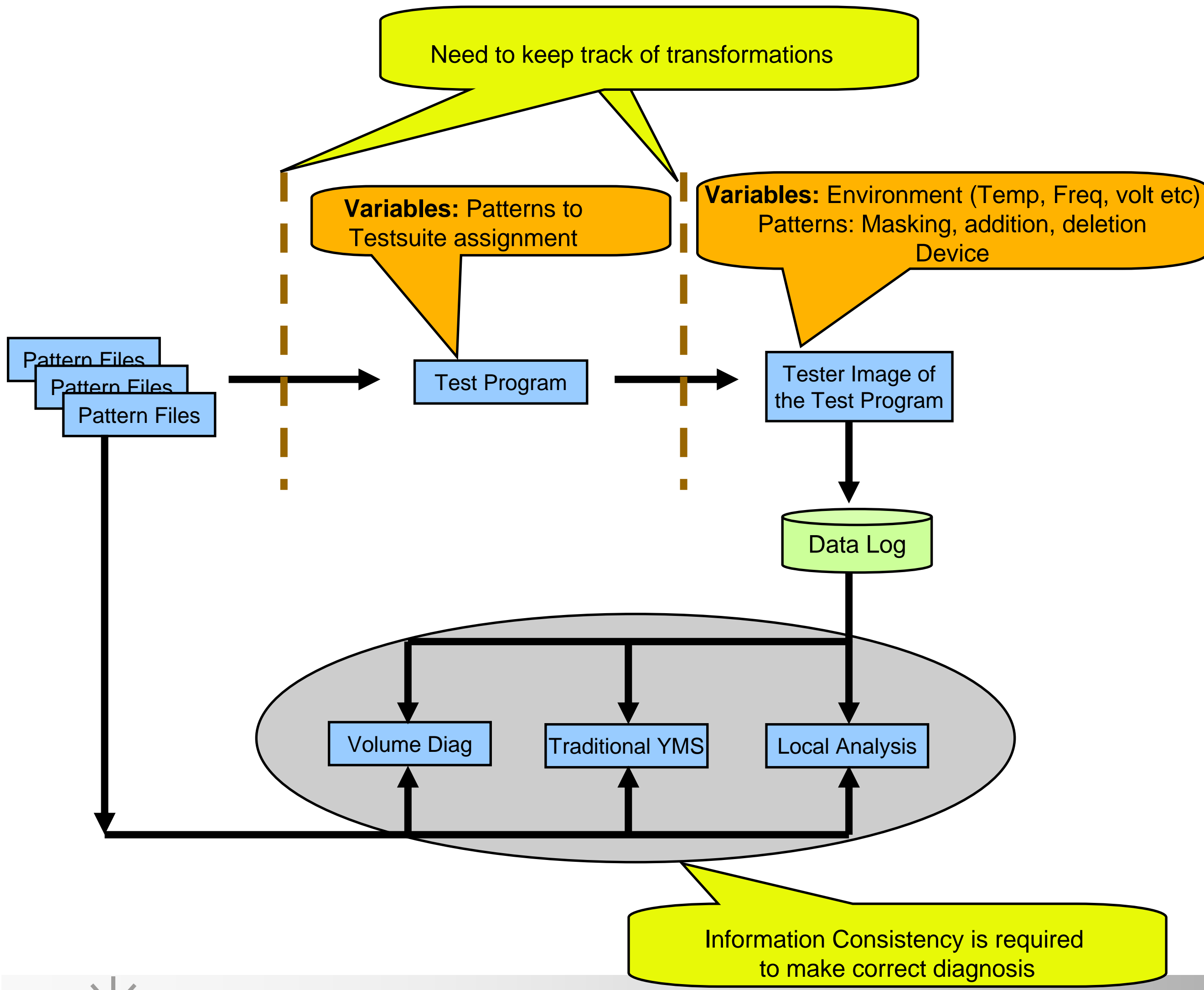
Conceptual Dataflow Diagram



Simple TestFlow



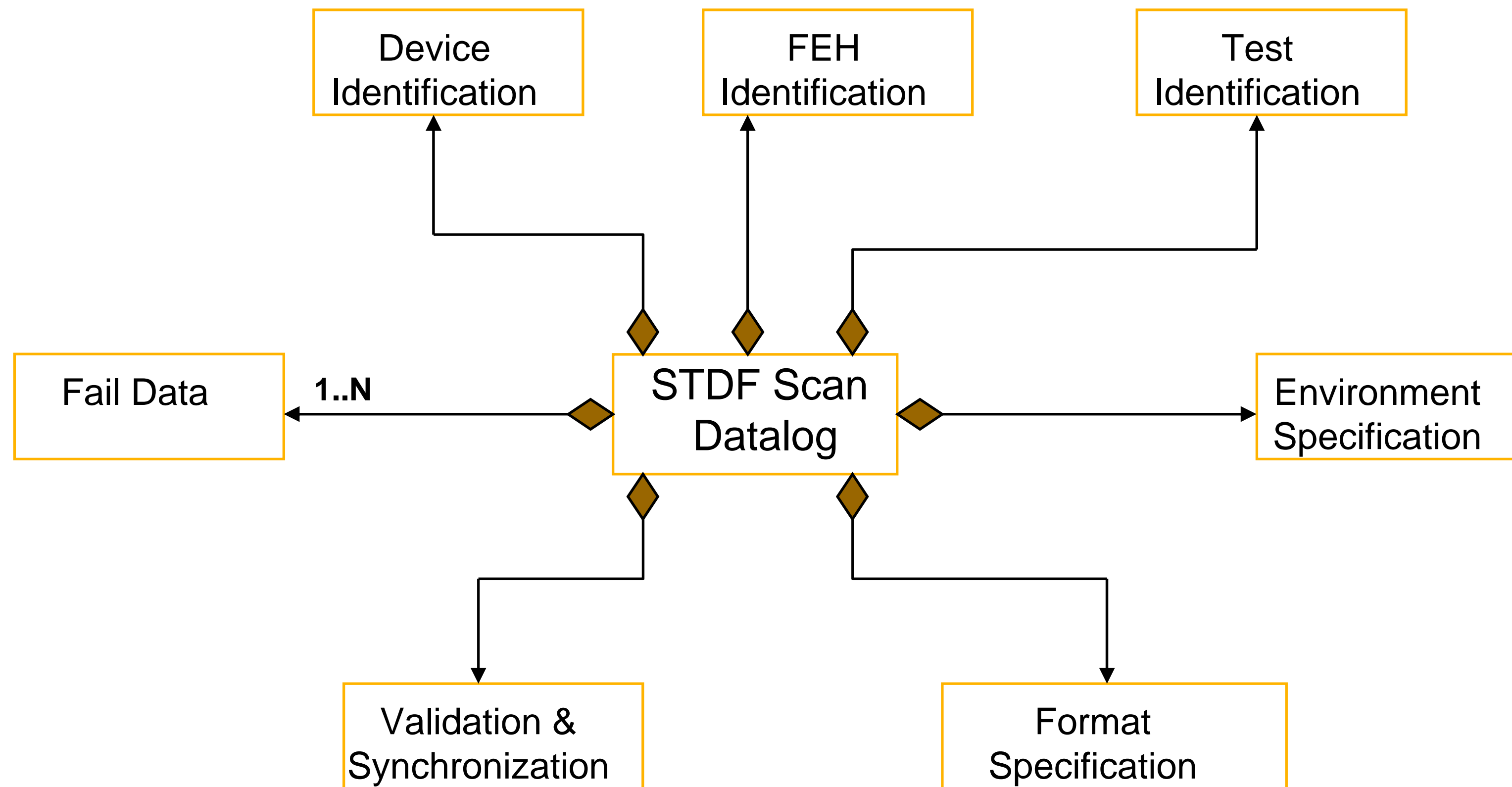
Conceptual Dataflow Diagram



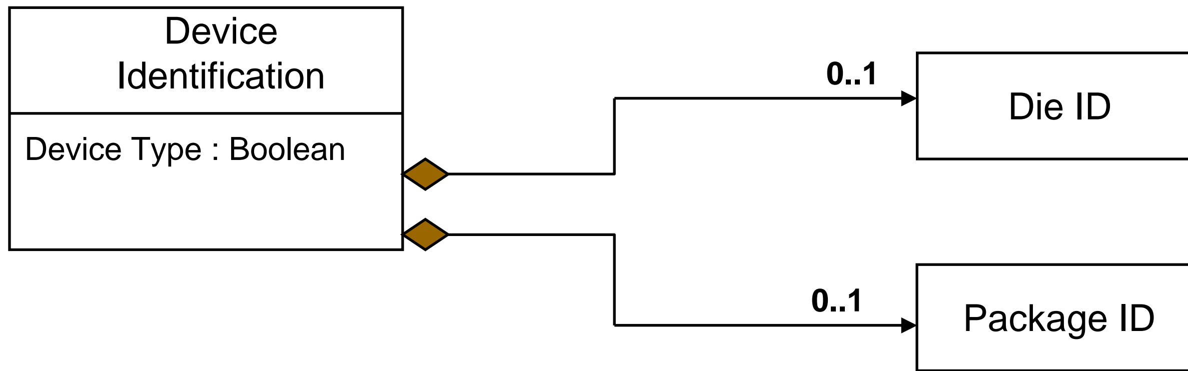
Datalog Format needs to provide infrastructure to communicate

- **Device Identification**
- **Test Identification (Test Flow, Test Suite, patterns)**
- **Test Environment (Temp, freq, volt, etc)**
- **Transformation information for synchronization**
  - **Assignment of patterns to test suites**
  - **Addition/deletion/truncation of patterns on tester**
  - **Any Name mappings**
  - **Buffer full/ datalog truncation**
- **Format Specification for Fail data**
- **Fail Data**

Scan Datalog Data Model



Device Identification



Die ID
WaferID : String
LotID : String
XCoord: Integer
YCoord: Integer
PartNumber: string
ElectronicID:string
OptionalField: Integer

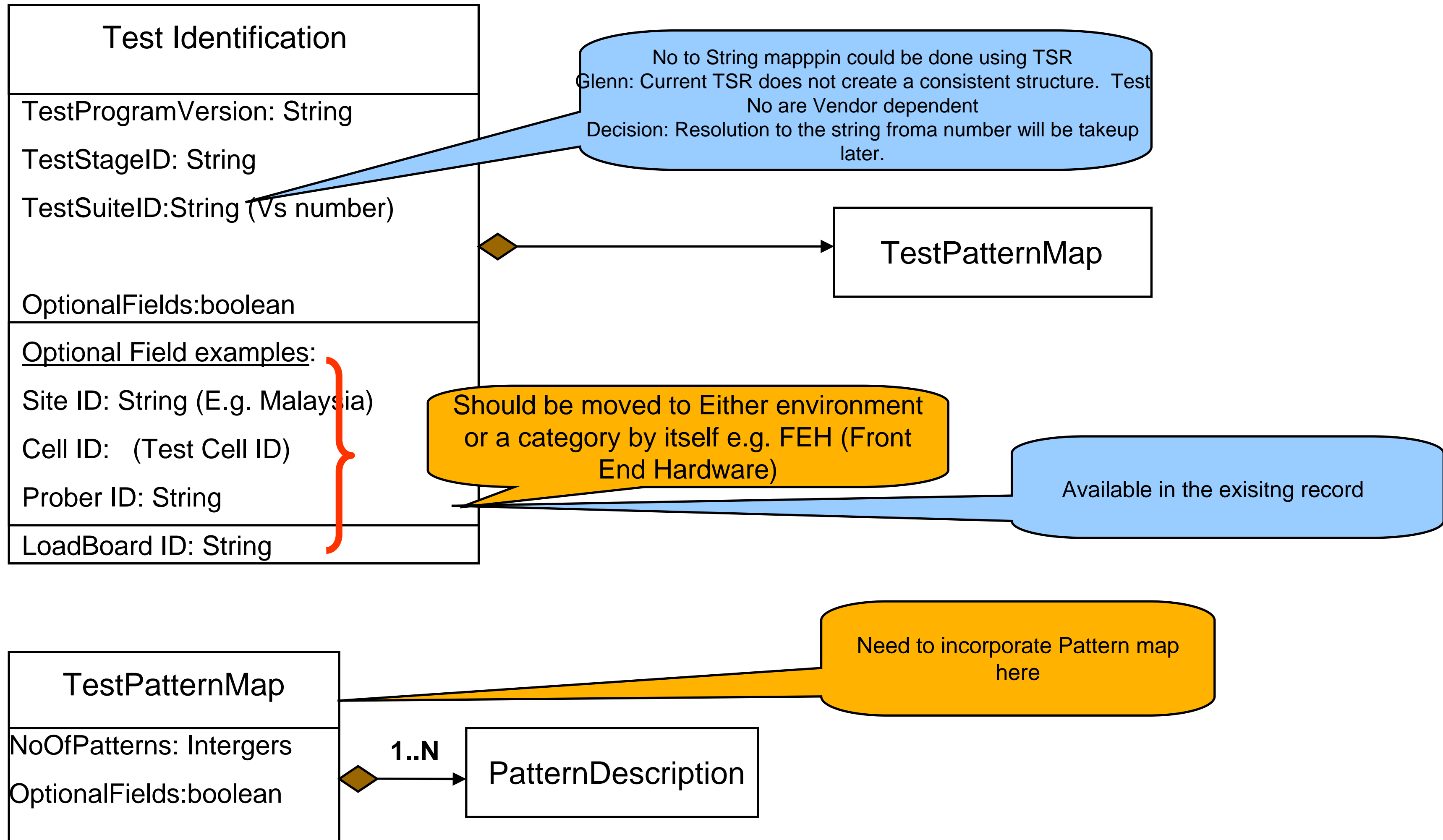
Package ID
SrNo : String
LotID: String
DieID: Integer
ElectronicID:string
OptionalField: Boolean

NOTE: Each Die in SiP will have a separate record

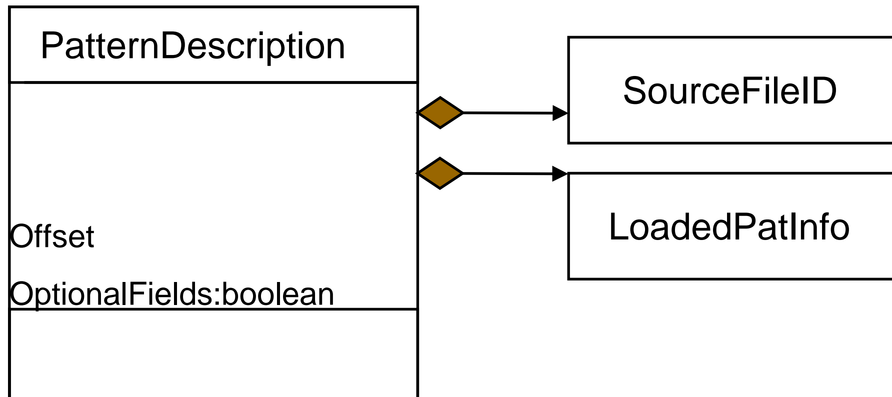
How do we obtain the Die ID?

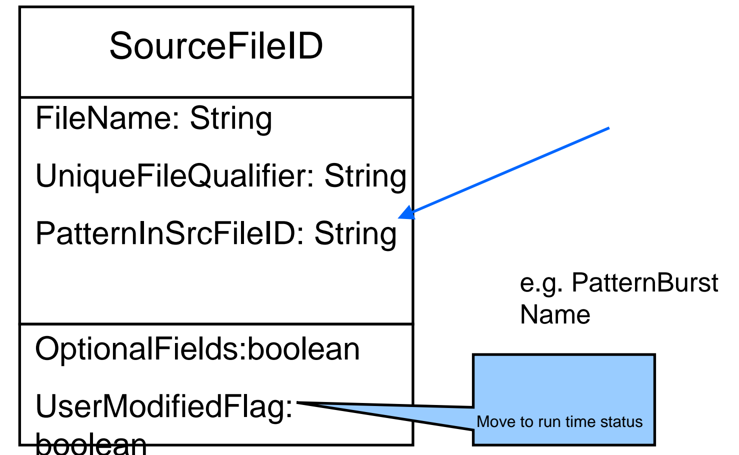
We will use the existing STDF V4 for identify the requirements

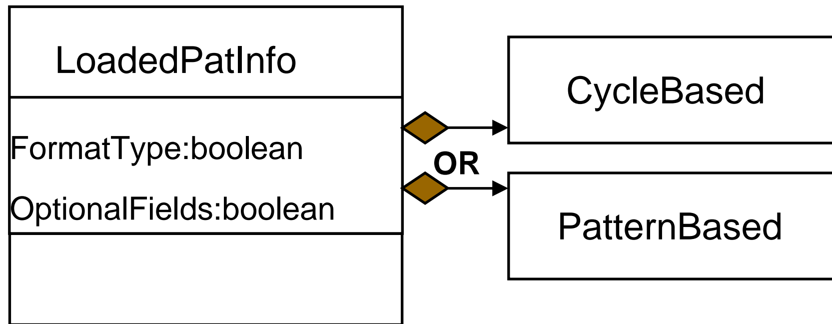
Test Identification

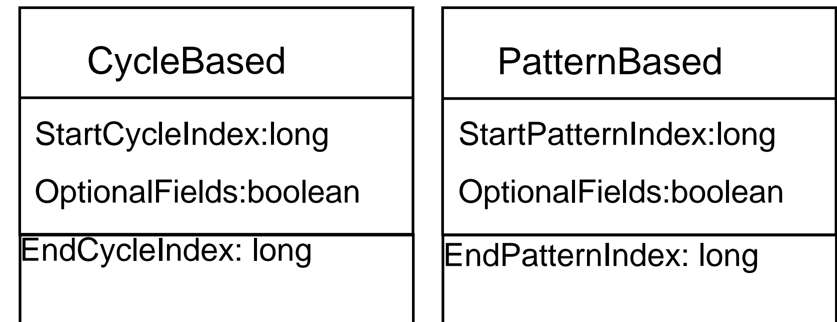


Test ID continued





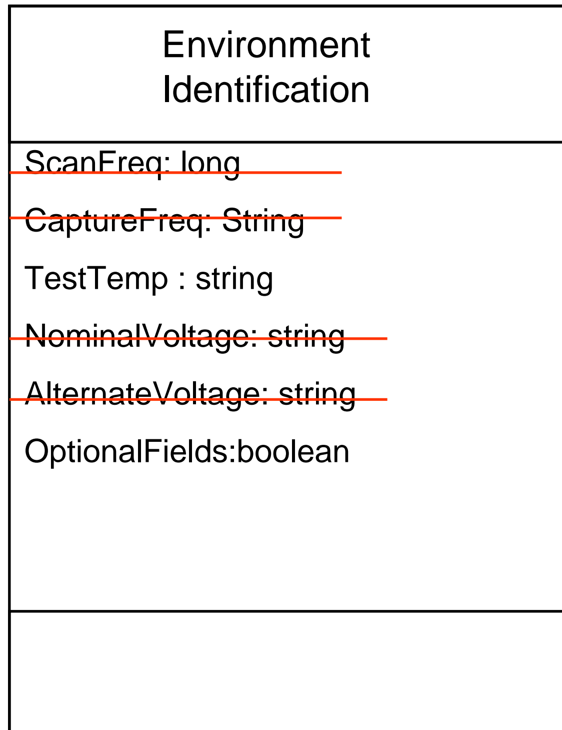


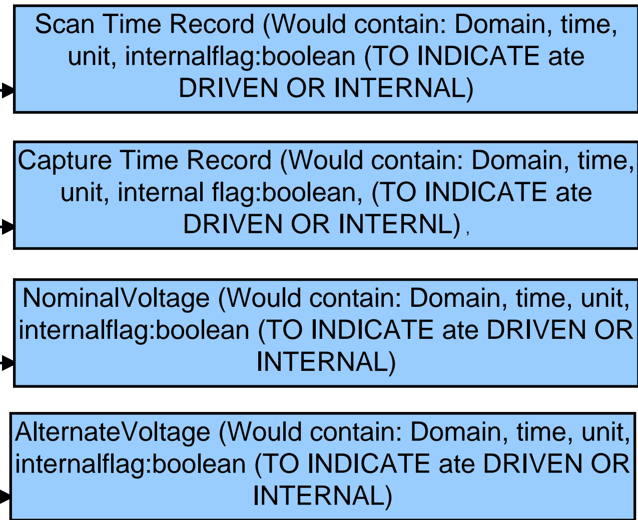


Dirty flag: (Decided not add); Existing fields can be used to indicate the dirty information.

Environment Identification

- ScanFrequency: Shift Frequency
- Capture Frequency: Functional speed
- Test Temperature
- Test Nominal Voltage
- Test Stress Voltage: To be used if any voltage stress in terms of bump or low voltage is applied
- User specific environment specification





•Do We need to store Shift Frequency?
•Do We need Frequency per pin?

- This data is meant to override the information in source pattern file
- The use model is that this info is available in the source file but any changes need to communicated.
- LOLS vs LOC indication…should be made part of the pattern source file.

Format Specification

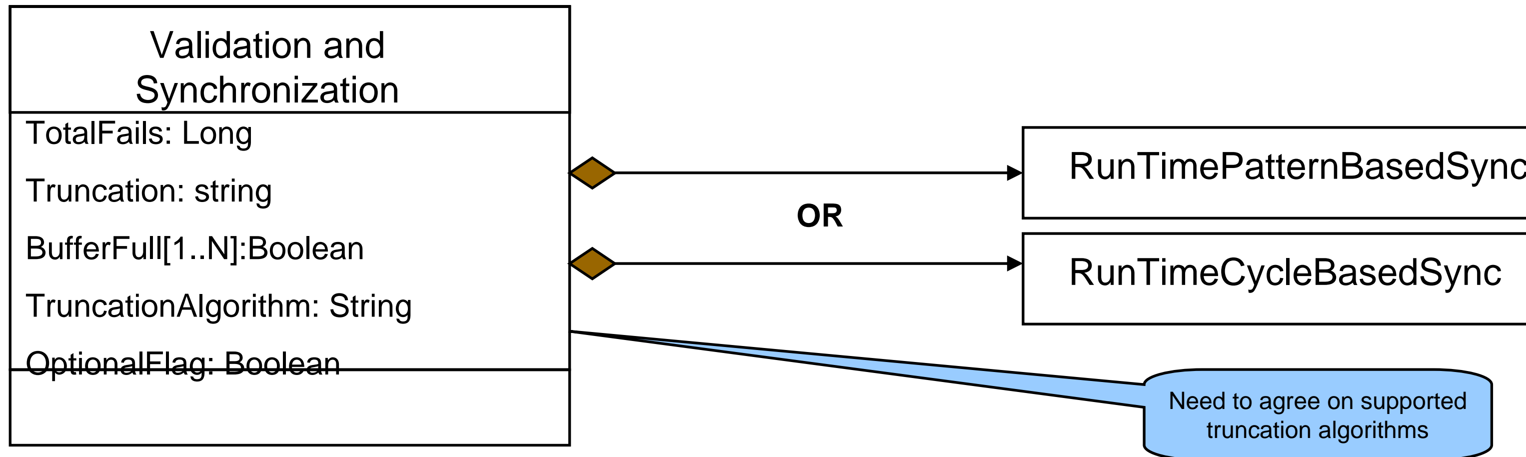
Format Specification
FailDataFormat: boolean
ExpectedDataAvailable:Boolean
ZHandlingFlag: Char
OptionalField: Boolean

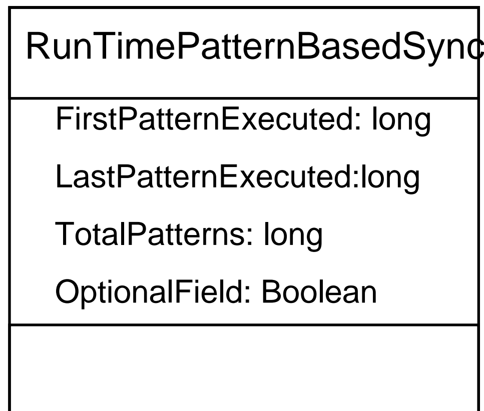
Add another enumerated values for not handled

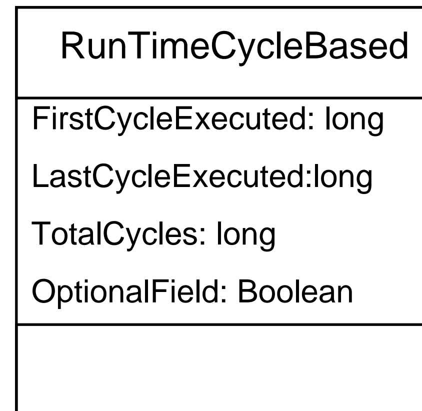
- Fail data format: ATPG (pattern based) vs. ATE (Cycle Based)
- Expected Data flag: (No expected data | with expected data)
- Z Handling flag: This is used to indicate how the Z states are handled and represented in the data log
- User specific

Fail data format to be a single filed with enumeration to indicate the the pairs for (FailData<Pattern/Cycle>, ExpectedDataAvailable<Yes/No>, DataCapture/DataDump<Yes/No> [Captured/measured data captured as string <Cycle no, captured data as string>])

Validation & Synchronization

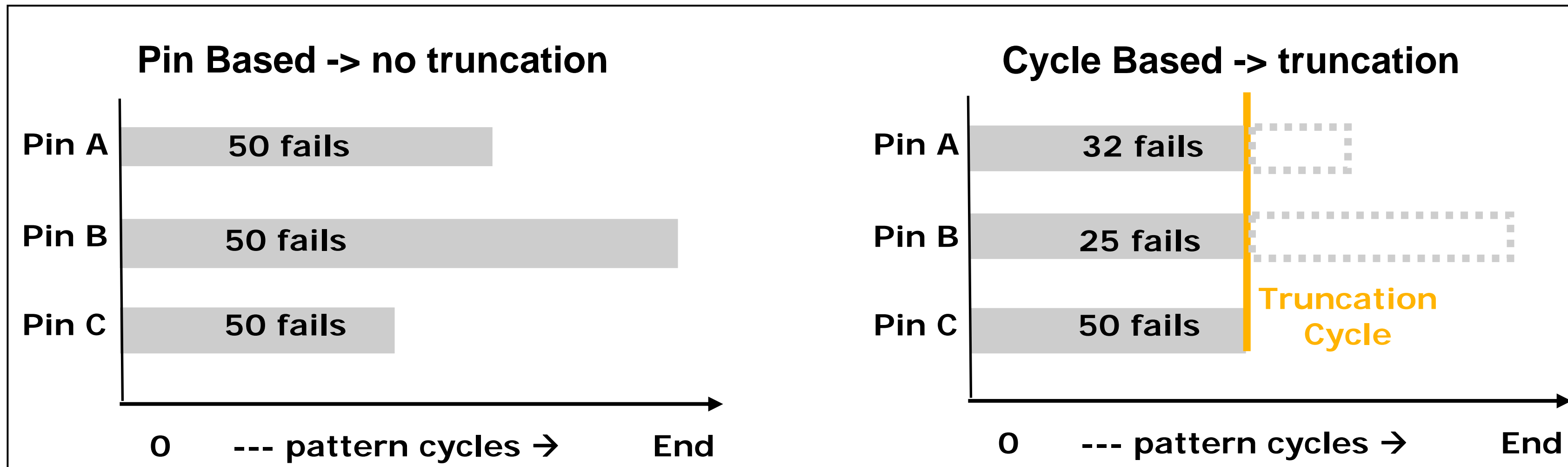




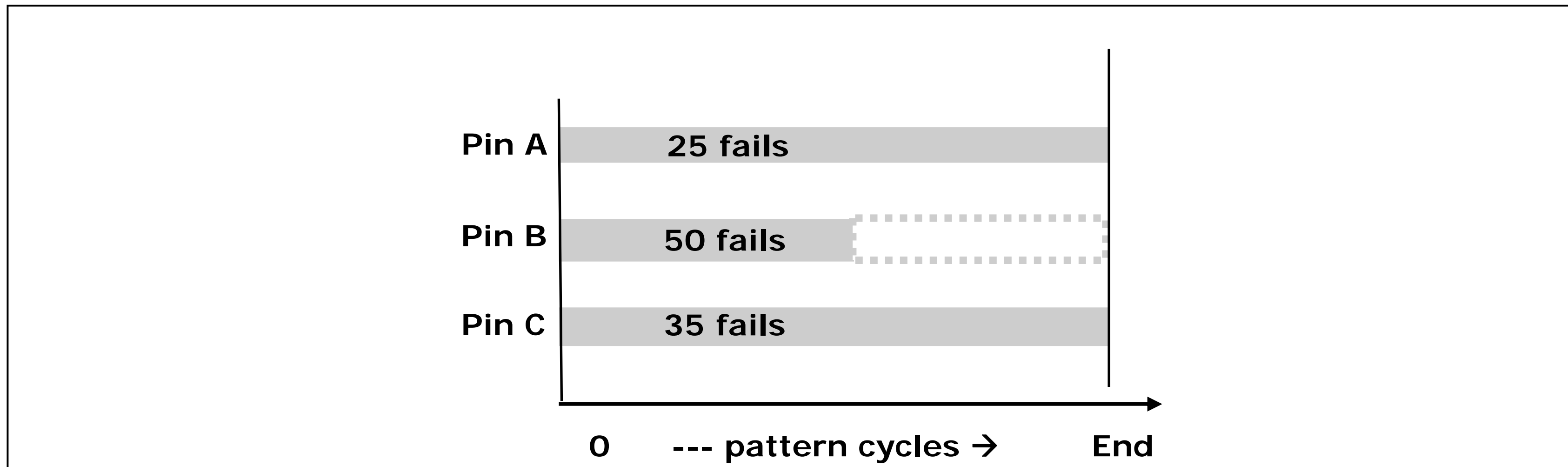


•Per pin v/s per cycle

Truncation Scenarios

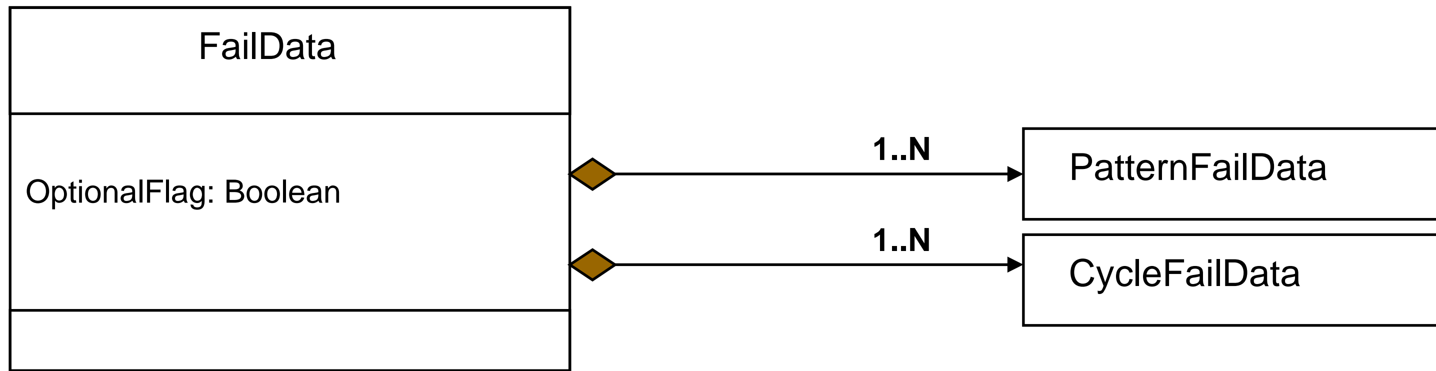


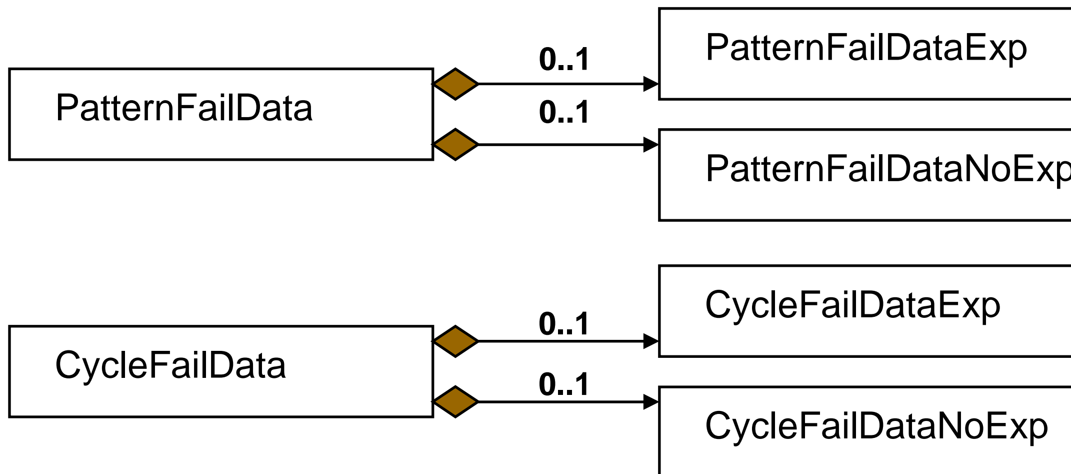
Buffer Full Scenarios



Fail buffer Size = 50 fails

Fail Data





Fail Data

PatternFailDataExp
PatterNum: long
OffSet: long
PinNum: Interger
MeasuredData: Char
ExpectedData: Char
OptionalField: Boolean

CycleFailDataExp
CycleNum: long
PinNum: Integer
MeasuredData: Char
Expected Data: Char
OptionalField: Boolean

PatternFailDataNoExp
PatterNum: long
OffSet: long
PinNum: Interger
MeasuredData: Char
OptionalField: Boolean

CycleFailDataNoExp
CycleNum: long
PinNum: Integer
MeasuredData: Char
OptionalField: Boolean

Additional Discussion Topics

- Merging data model objects, if there is overlap in contents or intent
- Example creation: Group to work on creating examples based on the discussions held so far.
- Single v/s multiple STDF file
- Self-contained vs reliance on other STDF records
- Various information classes and frequency of dumps
- Pin Name v/s Index
  - Pin Name Map
- Handling Xs
- Start cycle/pattern index (0 or 1)
- Getting core ID in a package

Category	Sub-Category	Sync point/Freq.
Device ID		Per device Device
FEH ID		Lot start?
Test ID		Lot start?
Environment		LotStart/per device
Format Spec		Per device
Validation/Sync		Per device
Fail data		Per device

VERIGY

END

Validation & Synchronization

Validation and Synchronization
F: Boolean ExpectedDataAvailable:Boolean ZHandlingFlag: Char BufferFull[1..N]:Boolean OptionalField: Boolean

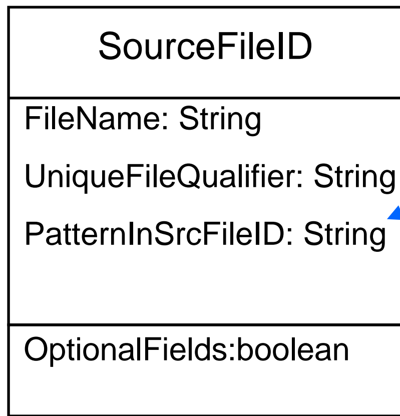
- First Pattern executed (Required only if fail data format is ATPG and first pattern is other than 0)
- Last Pattern executed (Required only if fail data format is ATPG and the last executed pattern is other than the last pattern in the memory)
- First cycle executed (Required only if fail data format is ATE)
- Last cycle executed (Required only if fail data format is ATE and the last executed cycle is other than the last cycle with entire pattern set)
- Total number of fails
- Total patterns (required only if the fail data format is ATPG.)
- Total cycles (required only if the fail data format is ATE.)
- Pattern 0 fail indication: This is set if there is any fail on any pin in pattern 0
- Buffer full identification: This information will identify for each pin whether there was a buffer overrun.

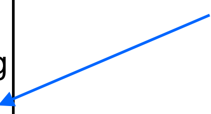
PatternBased
FirstPatternExecuted: long LastPatternExecuted:long TotalPatterns: long OptionalField: Boolean

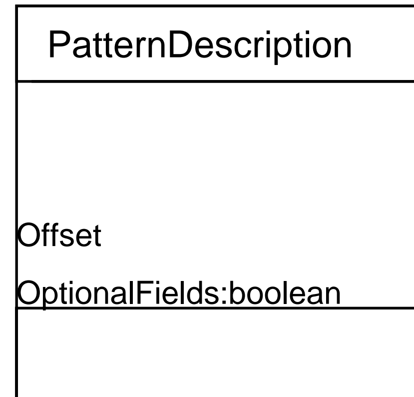
CycleBased
FirstCycleExecuted: long LastCycleExecuted:long TotalCycles: long OptionalField: Boolean

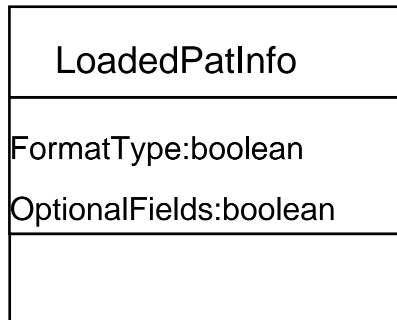
- External communication/press release
- Archiving decisions and proposal document format
- Continue discussions on requirements

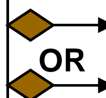
Test ID continued

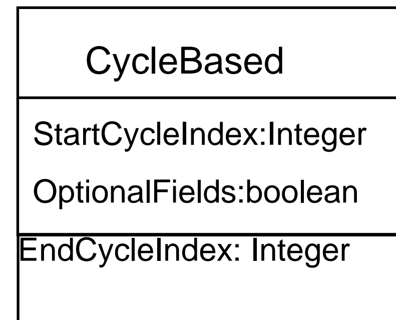


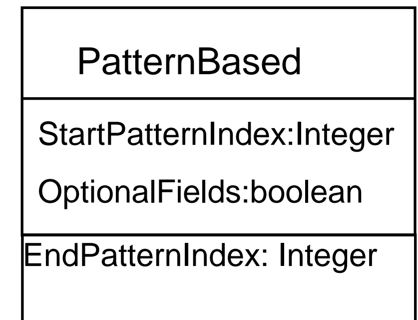












In the last meeting there was some discussion about reviewing the validation and synchronization aspect of the proposal. I have talked with our failure analysis team and this is one of their biggest concerns. Besides the issue of converting from a tester datalog to a diagnostics engine input format they routinely have issues of either incorrect atpg to ate file matching, or in most cases ate files have been modified.

Some of the modifications to the ate files that they have had to deal with are:

- 1-added initialization sequence to ate.
- 2-change in pin-name from atpg to ate.
- 3-added cycle in capture portion of pattern
- 4-removed cycle in capture portion of pattern
- 5-change of expect data to X in ate pattern

I believe that the first modification is being covered in the proposal, has the second modification been brought up before?

Modifications 3 and 4 could be debated if these should be allowed or handled. Most atpg tools now allow specifying the sequences used during capture which should eliminate modifying the pattern during translation to ate. But, there are legacy parts and flows that are slow to adapt. If support for legacy patterns is planned then we'll need to discuss support of this type of pattern modification.

The 5th modification should not affect the validation of the data since only failing ate values will be captured. But this could affect diagnostics as the diagnostic engine is assuming the expect value passed on the bad device, while on the tester it's unknown.

When we do review the validation and synchronization portion I would like to suggest we also determine what type of pattern modifications will be supported and which type will not be supported.

General Requirements

- Format should allow capture of millions of failures (what one would expect in 2010)
- Storing Environment information in the diagnosis records
  + allow storing diagnosis data in their own IT database
  - Duplication of data
  - consistency of keeping the replicated data consistent

Additional Issues

- Cycle count starting from 0 or 1: Decision- Use 0
- Multiple files in a test suite
  - Clarification on what is a test suite
  - One GDR v/s multiple GDR
- Test Mode Identification
  - What should we support?
  - Should it be part of the Test ID or Environment ID