

Key Element Definitions 3/10/2015

Table Of Contents

TPS Elements.....	2
Adapter Functional and Parametric Information (AFP)*	2
Diagnostic Data (DIAD)*	2
Diagnostic Services (DIAS)*.....	2
Digital Test Format (DTF) *	2
Maintenance Test Data and Services (MTDS)*	3
Master Conformance Index (MCI)*	3
Multimedia Formats (MMF)*	3
Prognostic Data (PROD)*.....	4
Prognostic Services (PROS)*	4
Test Program Documentation (TPD)*	4
ATE Elements	5
Data Networking (NET)*	5
Distributed Network Environment (DNE)*	5
Instrument Communication Manager (ICM)*	5
Instrument Drivers (DRV)*	6
Instrument Functional and Parametric Information (IFP)*	6
Receiver Fixture Interface (RFI)*.....	7
Resource Adapter Information (RAI)*	7
Resource Management Services (RMS)	7
Run Time Services (RTS).....	7
System Framework (FRM)*	8
Test Station Functional and Parametric Information (TSFP)*	8
UUT Elements	10
Design for Testability (DFT)*	10
Product Design Data (PDD)*.....	10
UUT Device Interfaces (UDI)*	11
UUT Test Requirements (UTR)*.....	12
Rules	13
Test Program to Operating System (TOS).....	13
Retired Definitions.....	13
Boundary-Scan Test Data (BTD).....	13
Computer to External Environment (CXE)	14
Diagnostic Processing (DIA)	14
Switch Functional and Parametric Information (SFP).....	14
Switching Matrix (SWM)	15

TPS Elements

Adapter Functional and Parametric Information (AFP)*

Adapter Functional and Parametric Data is a description of a test adapter's resource and path capabilities. The AFP provides a common, consistent and precise means for describing the test adapter data that facilitates the transfer of information between the providers and users of the data.

The AFP is critical to Framework Working Group objectives because of the high cost associated with ITA re-design when TPSs are moved from one platform to another.

Diagnostic Data (DIAD)*

Diagnostic Data is that information which supports the investigation and analysis of the cause or nature of a condition, situation, or problem through all phases of a system life cycle. Diagnostic data generally consists of models and procedures for determining the health state of a system.

Understanding this key element is crucial to the formulation of information models and standards that will support diagnostic information use and availability at and across all maintenance levels. A subdivision into three major categories describes the scope and identifies applicability of this key element:

1. Current test stand related data
2. Onboard and At-wing data
3. Evolving Integrated Diagnostics Data

Diagnostic Services (DIAS)*

Diagnostic Services are those standardized interfaces that allow transmission, conversion and retrieval of diagnostic data for the utilization in the maintenance process. These services link execution of a test with a diagnostic process that utilize test results and suggest conclusions or additional actions that are required.

Digital Test Format (DTF) *

Digital Test Format represents the data formats used to convey information used in conjunction with digital tests (e.g., test vectors, fault dictionaries) from digital test development tools to the test platform. This element is to be used for capturing the output of digital automatic test pattern generators, including patterns, full-fidelity timing, and levels.

The principal effort in transporting a digital TPS is focused around timing issues. The support of full-fidelity timing in a standard could reduce TPS porting time by an order of magnitude. For this reason the DTF was considered critical to Framework Working Group objectives.

DTF reduces cost associated with re-hosting a digital TPS. It provides the digital test data in a format that is readable directly by the ATS.

Maintenance Test Data and Services (MTDS)*

There is a need to share maintenance information across levels and across weapon systems. The current information structures are often service or weapon platform dependent. While these systems do an admirable job of providing a logistics infrastructure, they seldom share information back up the maintenance chain or across platforms and services. Given a standard set of definitions and data formats, information may be shared among all users who have implemented the standard forms.

MTDS enhances run time execution of the test program by capturing and using information developed during maintenance activities. Maintenance services are those standardized interfaces that allow transmission, conversion and retrieval of maintenance test data for the utilization in the maintenance process. A standard format for maintenance data can also be used in the design and development of future models of the system by placing constraints on the design engineer.

Master Conformance Index (MCI)*

The key element, Master Conformance Index (MCI), will provide configuration information and supporting resources required to test, evaluate and maintain a UUT. It will define and standardize on a common schema required to identify the configuration and locate the items of test program, Automatic Test Equipment and Unit Under test. Further this key element can be used to cover the product life cycle test and evaluation requirements from design verification to field support. This will support the Framework Working Group interoperability and reuse objectives.

Multimedia Formats (MMF)*

Multimedia Formats denote the formats used to convey hyperlink text, audio, video and three-dimensional physical model information from multimedia authoring tools to the end user. Application of MMF to test related information includes narrated video of test and repair operations, hypertext linked between test stations, TPS, fixtures and UUT documentation.

This element was deemed critical to the Framework Working Group objectives by reducing costs associated with TPS transportability. It provides a way to reduce reverse engineering efforts associated with test related information in the test program and the ATE.

MMF will reduce costs in any activity that requires user intervention. Adding the advantages of on-line documentation to ATS should increase the productivity of test personnel.

Prognostic Data (PROD)*

Prognostic Data is that information which supports the predictive element of diagnostics associated with the analysis of the time to failure of the component or system.

Analysis of the time to failure of a component or system can be achieved via test results, analysis and models from the UUT, information models from DIAD/ DIAS and MTDS elements are used with models and algorithms to assess degraded state of the device and predict failure progression.

Prognostic Services (PROS)*

Prognostic Services are standardized interfaces that facilitate transmission, conversion and retrieval of prognostic data for utilization in the maintenance process across all levels.

Test Program Documentation (TPD)*

Test activities exist within a larger context of a product life-cycle. At each phase of a product lifetime, testing is done to either prove that a specific instance of the product is good, or diagnose specific anomalies. Test Program Documentation is a part of the reference material that is applicable to the overall process; it provides information about the applicable test requirements satisfied by the test program, what is to be tested, how it shall be tested, the expected results, and the possibly corrective actions.

The TPD is critical to the framework working group objectives. The lack of standards results in ambiguities, inconsistencies and redundancies in documentation, which in turn can lead to duplication of test efforts over the products life cycle. Accurate and complete documentation improves TPS utilization, maintainability, interoperability, and rehost.

The TPD shall contain all documentation applicable to a test program. This could include the Test Requirements Document (TRD), Test Program source code, Test Program Instructions (TPI), and Test Diagrams.

To support the goals of interoperability and rehost, TPD shall provide a means to describe the TPS in a standard tester-independent electronic format that is easily accessible by test developers, system and component designers, and test operators.

ATE Elements

Data Networking (NET)*

The Data Networking element represents the hardware interfaces and application-independent data transmission protocols used by the ATS controller to communicate with remote computers over Local Area Networks or Wide Area Networks.

The Data Networking element contains a layered set of application-independent data transmission protocols, operating on top of a hardware layer. The following terminology is commonly used for the hardware and software layers: Physical layer, Data Link layer (sometimes called Network access layer or Network interface layer), Network layer (sometimes referred to as the Internet layer), and Transport layer.

The interface between the ATS controller and the physical layer is commonly provided through a network adapter, also called network card or network interface card (NIC). The data transmission protocols are commonly implemented in the Operating System (OS) of the ATS controller.

The NET element supports the operation of application-level protocols defined under the Distributed Network Environment (DNE) element.

Distributed Network Environment (DNE)*

DNE is the set of application-level protocols, software interfaces, data formats, etc. supporting the exchange of ATS-specific data between the ATS controller and remote computers over Local Area Networks or Wide Area Networks.

The DNE element facilitates transmission of test results, maintenance information, and other data from remote sites, management and distribution of software updates, remote access to ATS controllers, distributed testing scenarios, etc.

The operation of the DNE element is made possible by the hardware interfaces and application-independent data transmission protocols defined under the Data Networking (NET) element.

Instrument Communication Manager (ICM)*

The Instrument Communication Manager is the interface between instrument drivers and the software component that supports communication with instruments independent of the bus or other protocol used (e.g., VXI, PXI, Ethernet/LXI, IEEE-488.2, TIA-232, USB).

Historically, vendors of GPIB and VXI bus hardware provided software drivers for their buses that were different according to the hardware bus protocol being used. The same functions of the same instruments were not accessed through software in the same way across buses and host platforms. For example, different manufacturers of GPIB cards had proprietary and unique software calls. This scenario impedes the transportability of test programs from one platform to another. For this reason, the ICM element was deemed critical to Framework Working Group objectives.

A standardized ICM enables higher level software to be interoperable and portable between vendors and across different platforms. The ICM allows instrument drivers to be ported across test systems, and therefore, instruments can be moved with the test program if the instrument's functionality does not exist on the target system. Each of these items improves the interoperability of test software and the ability to re-host test software from one test system to another.

Instrument Drivers (DRV)*

An instrument driver is a software component that handles the details of controlling and communicating with a specific instrument. The instrument driver includes all the communication details of a particular instrument in high-level software functions that are directly usable by software components from the Resource Management Services (RMS) element. This relieves these software components from having to include low-level I/O commands for communicating with an instrument.

This component of an ATE was deemed critical because of its impact on TPS transportability and instrument interchangeability issues.

The use of instrument drivers instead of low-level I/O commands simplifies the software development or rewrite required for instrument replacement, technology insertion and TPS rehosting. It reduces the training time required for personnel who construct and use instrument drivers by defining a standard method for software control of test instruments. An accepted industry standard assures users that instrument drivers from multiple vendors are designed, packaged and used in a consistent way. Furthermore, driver standards facilitate the creation of general-purpose software tools for driver configuration and support.

Instrument Functional and Parametric Information (IFP)*

Instrument Functional and Parametric Data is test related information and data formats used to define what an instrument can measure, stimulate, and/or load the circuits to which it is attached. The IFP will be used by the TSFP to describe the test stations resource capabilities.

The IFP will reduce cost associated with TPS re-host activities by providing a common description of test station instruments which will be utilized by the TPS development environment and run time execution when directing test actions.

Receiver Fixture Interface (RFI)*

The Receiver and Fixture are mechanical subassemblies used to align and house mating contacts to form the Receiver and Fixture Interface (RFI) system. The RFI provides for interchangeability of mechanical, electrical, receiver, fixture, connector product assemblies from various manufacturers under an open architecture.

The RFI should be flexible to allow system enhancement; and minimize fixture costs through a range of scalability options (different fixtures sizes can be mated with one full-size receiver).

Resource Adapter Information (RAI)*

RAI defines a set of rules that describe the content and format for test requirements implemented in configuration managed test program source code. Test program source code consumers confirm compliance with the RAI rules in order to maximize test program transportability

Resource Management Services (RMS)

The collection of software that resides between Test Programs and Instrument access layers, which provides one or more of the following:

1. Allocation of test station assets,
2. Path determination for signal routing,
3. Path connection using switching,
4. Conversion of test procedure action to instrument access layer communication,
5. Path loss compensation,
6. Timing constraint assurance,
7. Assets utilization to determine its maintenance action (health),
8. Connectivity assurance.

Run Time Services (RTS)

Run Time Services include software services needed by a test program and not handled by services supplied by other Framework elements. Examples of such would include error reporting, data logging, and input/output functions.

This element is critical to Framework IPT objectives because of the impact on TPS transportability. Standardizing a set of run time services will facilitate the

transfer of TPSs from one platform to another by reducing reverse engineering costs normal associated with that activity. This is accomplished by using standard layers to insulate the test program from specific operating systems features or hardware platforms. Without a standard run time interface, service requests from test programs not available on a target system would require extensive test program modification, and therefore, costs.

These runtime services required are provided in several domain specific areas
Error Reporting - TPSs and environments should utilize the underlying OS Event logger for platforms such as Windows and Linux

File I/O functions - should use the OS file system calls for Windows & Linux as these are well documented and emulated across platforms.

Graphical Operator Messages - TPSs should utilize HTML presentation format

System Framework (FRM)*

The ATE System Framework element defines a collection of software interfaces and information formats that provide abstractions to control and communicate with the ATE. The software interfaces include support services to manage the configuration, creation and runtime support of instrument resources, as well as any additional abstractions to provide common control interfaces. The levels of abstractions for a generic FRM are defined by the key elements associated with the ATE. The choice of system framework for an ATS will enable interoperable components adhering to that framework to be configured together to help provide the higher functions of a test system. Examples of such frameworks would provide abstractions for signal control, Synthetic Instrument, or LXI instrument Synchronization.

System framework reduces cost by allowing interoperable components to be added, and provides higher level interfaces for abstractions that support higher degrees of interchange between TPS and ATEs.

It is envisaged that there may be several FRM abstractions, which rather than build on each other, can exist in parallel, to support different TPS environments. These additional levels of abstraction's effectiveness will be dependent on how well they meet the ATS framework definitions procedures.

Test Station Functional and Parametric Information (TSFP)*

Test Station Functional and Parametric Data is a description of a test stations resource and path capabilities. The TSFP will utilize the instrument signal capability descriptions from the IFP and in addition provide a description of test station connectivity to allow the routing of these signals to the test station interface.

The TSFP provides a common, consistent and precise means for describing test station data that facilitates the transfer of information between the providers and

users of the data. This information describes test station capabilities required to support various use case applications such as TPS rehost.

The TSFP shall be defined in a way that will allow resource and path data to be used for resource management and path allocation. The TSFP can also be used to compare a UUT's test requirements to a candidate test system's capability for purposes of validating test requirements on the targeted test station.

UUT Elements

Design for Testability (DFT)*

In today's complex systems, it is important that test and design functions are integrated during a product's development. Standards for defining and communicating this information will ensure reuse of essential maintenance and diagnostic data during a product's life cycle. Design for testability corresponds to the capabilities designed into a component or system that facilitate testing. These capabilities are determined based on test criteria derived from the tradeoffs made during the design and manufacturing phase of a product. DFT impacts the design of a product and directly affects the manufacturing, operational, and maintenance phases of a product's lifecycle. It can also affect the reliability of the product since it often introduces additional components that can fail.

Design for testability is considered critical in this framework because it improves the testability, diagnosability, and maintainability of the product. Diagnostic tools such as reasoners can be used more effectively on systems that have been designed for testability. DFT directly affects the, accuracy, quality, and ability to detect and isolate faulty components, which results in improved usability, availability, and operational readiness. As systems and their sensors become more complex the need for DFT becomes more critical. DFT can reduce maintenance actions associated with resolving ambiguity groups through the use of Built in Test data (BITD). DFT can also reduce testing time by utilizing tools such as "directed TPS". In a complex dense system, DFT is essential for detecting embedded failures.

DFT can be implemented by incorporating additional software and / or hardware added to the unit and / or the environment that the unit operates in. Some common forms of DFT are Built In Test (BIT), Built in Test Equipment (BITE) and Built In Self Test (BIST). Boundary scan is a component level example of BIST. BIT, BIST and BITE can be combined to increase the effectiveness of DFT. Much of the DFT efforts have addressed the IC / component level, and more work needs to be done at the system level and analog testing areas.

Product Design Data (PDD)*

Product Design Data is information that originates in the design process and which is needed for the development and sustainment of test and diagnostics. PDD includes information about structures that are present in the product solely or principally to support test, diagnostics and repair, rather than manufacturing.

PDD supports the reuse of data from the design phase rather than the recreation of it at the test development phase. It facilitates the transfer of information from CAD workstations to the TPS development, reducing errors and development

time. PDD contains UUT design information, such as components, sub-components, interconnections, and design considerations such as component failure rates. The PDD also identifies UUT faults and failure, and built in test (BIT) codes used for self diagnosis.

The PDD information considered critical in this Framework because of the potential impact it has to improve the quality of diagnostics during test and repair actions. Also PDD working with diagnostic tools can reduce test and repair actions by starting the test program further along in the process. This is sometimes referred to as a “directed TPS” which will start its testing at different places depending on symptoms or other input information. The monitoring of PDD can help identify “bad actors” or incipient failure modes as well as prognostics.

PDD BIT codes and faults & failures will improve run time execution of test programs by providing guidance to the diagnostic services within an ATS. In addition, during TPS development, candidate BIT requirements can be evaluated by contrasting the impact on design and production against maintenance and diagnostic test. Cost effective BIT requirements can then be imposed as design constraints.

PDD supports the back-annotation of test and maintenance information into the design environment, reducing sustainment costs.

Tremendous effort is currently spent in TPS development to reverse-engineer product information that was known to the designer but not forwarded to the test engineer. Costly errors are introduced when the reverse-engineering fails. The Framework Working Group has identified PDD as an interface between design and test that eliminates these costs.

UUT Device Interfaces (UDI)*

The UDI allows for standardized testing of particular types of UUT technologies. The scope of this key element is to provide a standard mechanism for developing and exchanging reusable software libraries applicable to the testing of common UUT technologies, where the hardware interfaces associated with these libraries, test connectors, cables and fixtures, would be part of their UUT test requirements (UTR). These technologies and their means of interfacing utilize test standards that can be used in the design and development process.

Examples of standardized technologies are digital radio, wireless communication, switching, fiber optics and networking which are being employed in existing as well as new UUTs. Examples of the associated UDIs are signal libraries together with test descriptions describing parameterized test methods and performance verification for communication with devices that have limited accesses but highly complex inputs and outputs.

Examples of communication interface requiring standardized test methods could be 1149 (Boundary Scan), WI-FI, 802 (LAN/MAN), Ethernet, USB and other busses such as 1553, 429 etc.

The software would consist of standard tests (or library of tests) applicable for that UUT type to determine performance or functionality e.g. Radio tests (SINAD, Sensitivity) Radar Tests (Doppler, Reply Efficiency) EW testing; Navigation Aids (IFF, TACAN).

DUI will have an impact in the following areas: Interfacing the UUT to the test fixture, standardized reusable test method and procedure libraries. These test and evaluation requirements, when adapted to the operation and support of the product could significantly reduce the test cost and foot print of support items and in the case of the TPS enhances rehost.

UUT Test Requirements (UTR)*

UUT Test Requirements include the information used to define to the test environment the load, sense, and drive capabilities that must be applied to the UUT to test it, including the minimum performance required for a successful test.

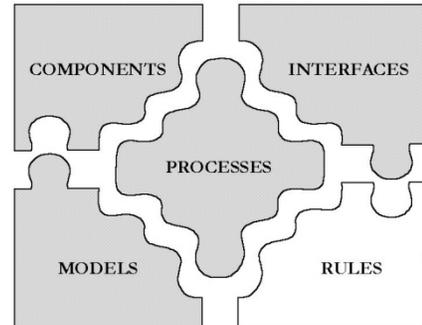
The test development process begins with a set of requirements. Some of these requirements address the development process itself. An example would be a requirement for a certain level of documentation or for the use of a particular test system. Other requirements address the diagnostic portion of the test. A common example is a requirement for a minimum level of fault detection or isolation. The remaining requirements are test requirements, and these establish constraints on the final test program.

A clear understanding of UUT test requirements is critical to any TPS re-host effort. For first-time TPS development, the lack of formal test requirements is sometimes offset by support from the product designers. For TPS re-host, however, such support is unavailable and the test engineer must reverse-engineer the test requirements from the existing TPS. This historically has been difficult because the TPS is a mixture of test requirements and implementation decisions, and these must be separated. A re-hosted TPS must obey the test requirements but is free to ignore the implementation decisions.

Rules

Test Program to Operating System (TOS)

The CIWG defined the TOS element as calls to the host operating system made directly from the test program. It was recognized that direct calls to the operating system from the test program had major impact on the interoperability and re-hostability of the TPS between platforms. The ARI re-examined this problem and believed the general thought applied across all implemented interfaces in a system. Therefore the TOS element was eliminated and replaced with a general rule.



By enforcing a rule that implemented layers of an ATS may not be bypassed by direct communication to another interface or layer in the architecture facilitates the interoperability and re-hostability of the TPS between platforms.

Retired Definitions

Boundary-Scan Test Data (BTD)

Boundary scan is a method for testing interconnects (wire lines) on printed circuit boards or sub-blocks inside an integrated circuit. Boundary scan is also widely used as a debugging method to watch integrated circuit pin states, measure voltage, or analyze sub-blocks inside an integrated circuit.

Boundary Scan Test Data in the information required for an Integrated circuit board using On-Chip infrastructure to test the correct operation of its internal devices

Boundary Scan Test Data can serve as a trigger for later maintenance actions, often taking test during operations or in environments that cannot be duplicated at or transferred to later maintenance levels.

Boundary scan on components and test busses on circuit cards are classic examples of modern test methods that can greatly help the test engineer.

To provide the boundary scan capability, IC vendors add additional logic to each of their devices, including scan cells for each of the external traces. These cells are then connected together to form the external boundary scan shift register (BSR), and combined with JTAG TAP (Test Access Port) controller support comprising four (or sometimes more) additional pins plus control circuitry.

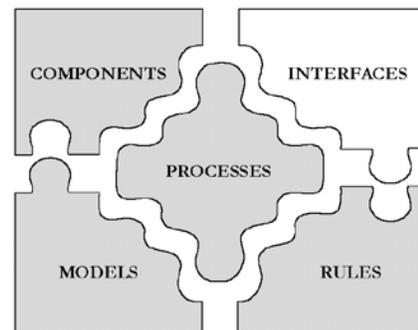
Boundary Scan is not just test information but rather designed features of the product and a 'test program', which allow greater access and control to the product than would otherwise be present.

BTD is critical to the framework, as it provides in-depth component testing through a small single interface, providing a reduced maintenance logistic footprint and improved diagnostic capability in a modern test programming environment.

Computer to External Environment (CXE)

Computer to External Environments describes the communication methods between a host ATS and remote systems.

The CXE was identified as a key element because standardizing it is expected to reduce the cost of transferring information during re-host of a TPS. A standardized format for transmission of data from remote sites facilitates several areas of improvement. Software updates can be managed and distributed through this element. Distributed testing is feasible by directing and obtaining test-related data through this element. It works together with NET to achieve these benefits.



Diagnostic Processing (DIA)

Diagnostic Processing is the interface protocol linking execution of a test with software diagnostic processes that analyze the significance of test results and suggest conclusions or additional actions that are required. The DIA component was deemed critical because of its potential to reduce test costs and facilitate the reuse of test-related information.

The ARI refined the definition of the CIWG's DIA key element by subdividing it into two elements. Diagnostic Data (DIAD) is an information model in this architecture and Diagnostic Services (DIAS) is an interface in this architecture. Each is explained in the following section.

Switch Functional and Parametric Information (SFP)

Switch Function and Parametric Data is the information and formats used to define the interconnect capabilities of the switch matrix, how these capabilities are accessed, and associated performance parameters.

An adequate description of switching capabilities is necessary to TPS transportability issues. Use of the SFP should facilitate reuse of information

related to switching during TPS re-host activities, noticeable as a reduction in reverse engineering costs for this type of activity.

Switching Matrix (SWM)

The Switching Matrix is a hardware element description of the switch paths that connect ATS test and measurement instruments to pins on the RFX. It is a hardware component in this Framework. The SWM must also work with the RFX element.

To remain upward compatible, the SWM must be designed in building blocks that can be duplicated to meet worst case requirements. This facilitates a modular framework design that permits ATS integrators to incrementally augment their systems through add-on/duplicative features. This allows them to meet worst case requirements while maintaining downward compatibility with any smaller I/O increment.

This simplifies ITA design, reduces TPS costs, and places the switching under control of the ATE system software. The SWM works with the RFX to reduce costs associated with TPS transportability.