Lean Software Architecture for Automated Test Equipment

Adem G. Aydin

March, 2014

Outline

- Test and Characterization in Microelectronics
- Cost versus Reliability
- Automated Test Equipment
- SW/HW Co-design
- Proposed SW Architecture

Test and Characterization in Microelectronics

- Complexity of design requires through initial characterization.

- Cost/reliability (mission safety) of the system such components constitute requires functional test.

- Functional test is to create the real electrical working conditions for the components.

Cost versus Reliability

- What constitutes the cost of the functional test?
- How much can we spend on functional testing?
- Reliability concerns?

Automated Test Equipment

- Equipment: from simple DVM to more sophisticated mmWave tester.
  - A mmWave tester may include high frequency digitizer, frequency converter, rf sources, DC supplies, digitial i/o, spi generators, AWGs etc
- Automation: No human interaction is expected.
- Designed for “general purpose” and programmed for specific product and/or test requirement.

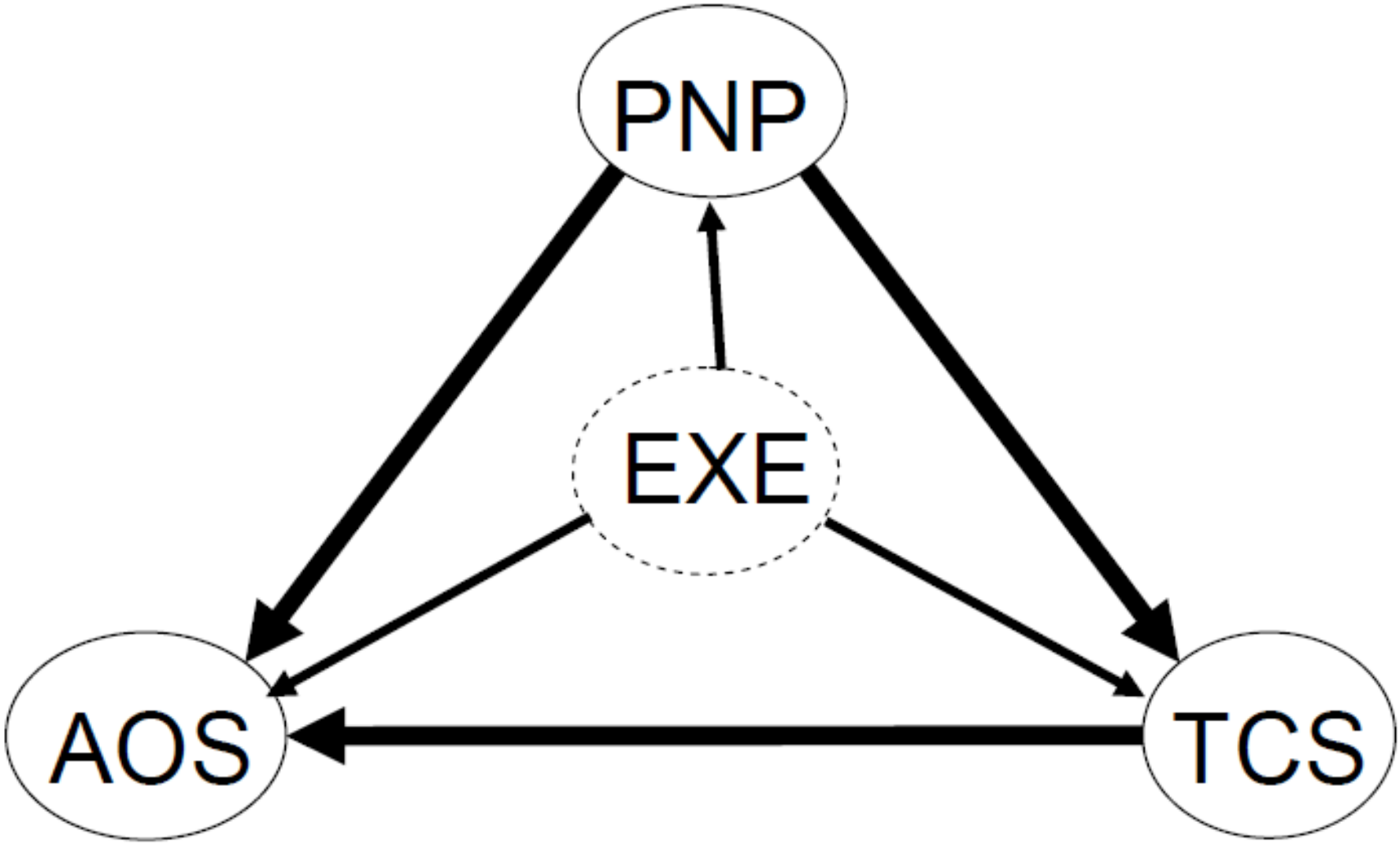
SW/HW Co-design

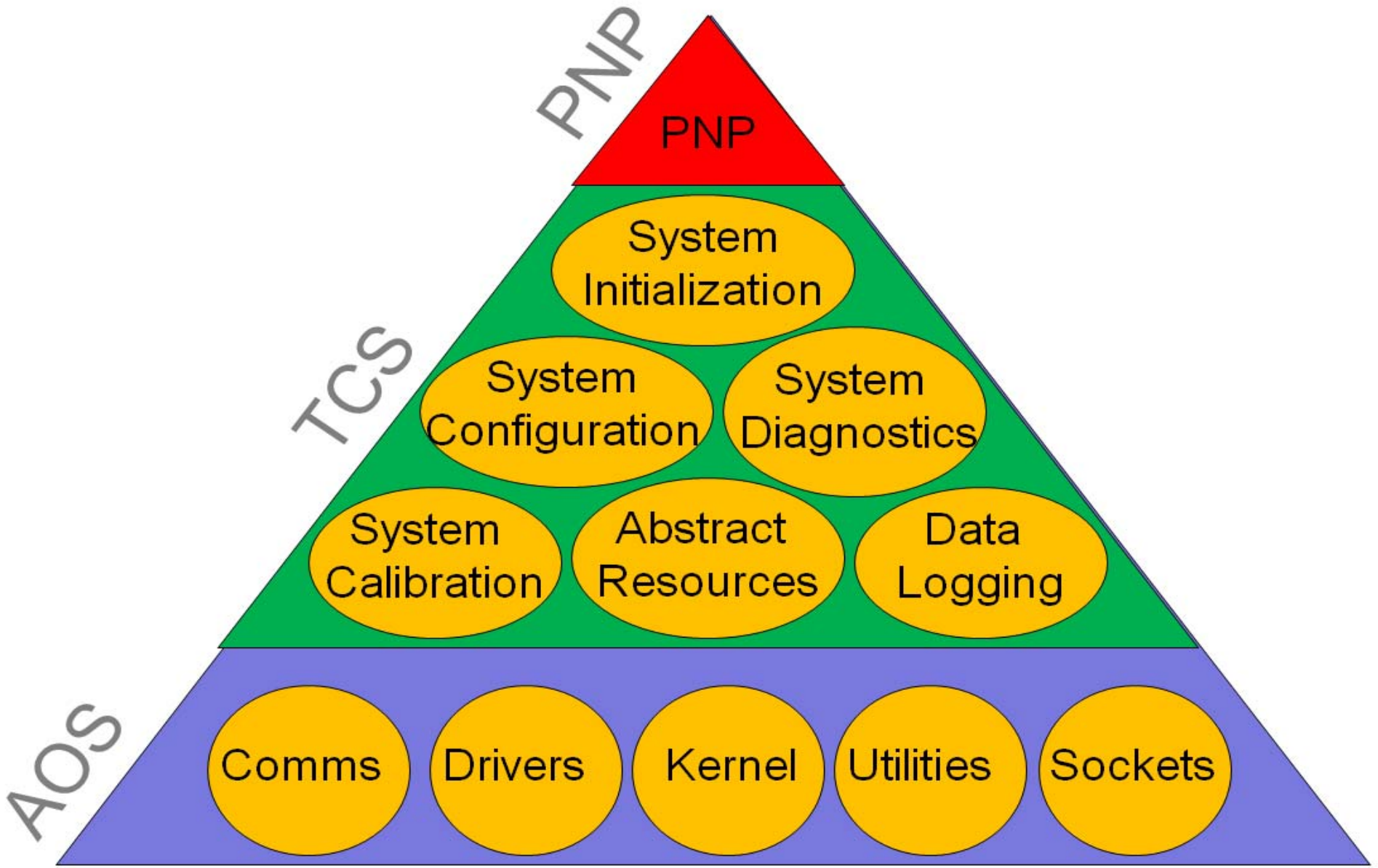
- SW/HW co-design is a “must”.
- ATE is as good as its SW && HW.
- HW must conform to somewhat generic usage w/o giving up additional features.
- SW must make all the features implemented by the HW available for use in a modular fashion.

Challenges

- Components and systems with new features
- New features require new test capabilities
- New test requirements result in
  - Software upgrades
  - Firmware upgrades
  - Hardware upgrades
- Modularity is essential for “allware.”

Proposed SW Architecture




PNP
PNP
TCS
System Initialization
System Configuration
System Diagnostics
System Calibration
Abstract Resources
Data Logging
AOS
Comms
Drivers
Kernel
Utilities
Sockets

Architectural Components

- AOS: Auxiliary Operating System
- TCS: Tester Configuration System
- PNP: Part Number Program
- EXE: Executable

Architectural Components

- AOS
  - Kernel
  - Communications
  - Utilities
  - Sockets
  - Drivers

Architectural Components

- TCS
  - Configuration
  - Calibration
  - Diagnostics
  - Abstract Resources
  - …

Architectural Components

- Abstract Resources
  - Voltage Source
  - Current Source
  - Voltmeter
  - Digitizer
  - …

Architectural Components

- PNP
  - Uses Abstract Resources from TCS
  - Agnostic of available hardware, except the limitations
  - Uses AOS for OS related tasks.

Architectural Components

- EXE
  - Executes system level functionalities
  - Provides interoperability for other programs, e.g., sockets
  - Communicates the options to the user

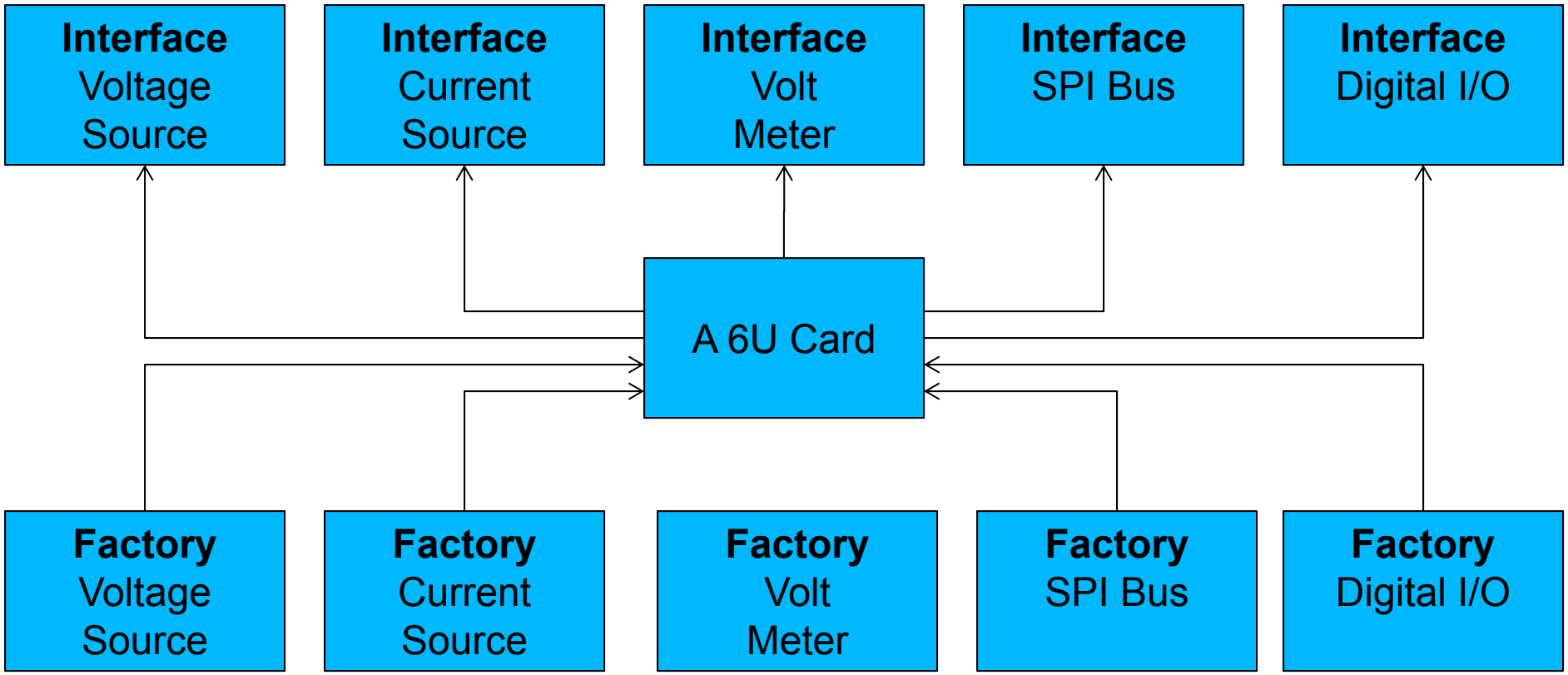
Instrument Control vs Measurement Platform [1]

- Instrument Control: What can you do?
  - Long learning curve
  - Lack of interoperability
  - Instrument dependency
  - …
- Measurement Platform: Can you do this?
  - Short learning curve due to standard definitions of abstract resources
  - Interoperability, new instruments can be easily integrated.
  - Instrument independent: if two different instrument provide the same functionality
  - …

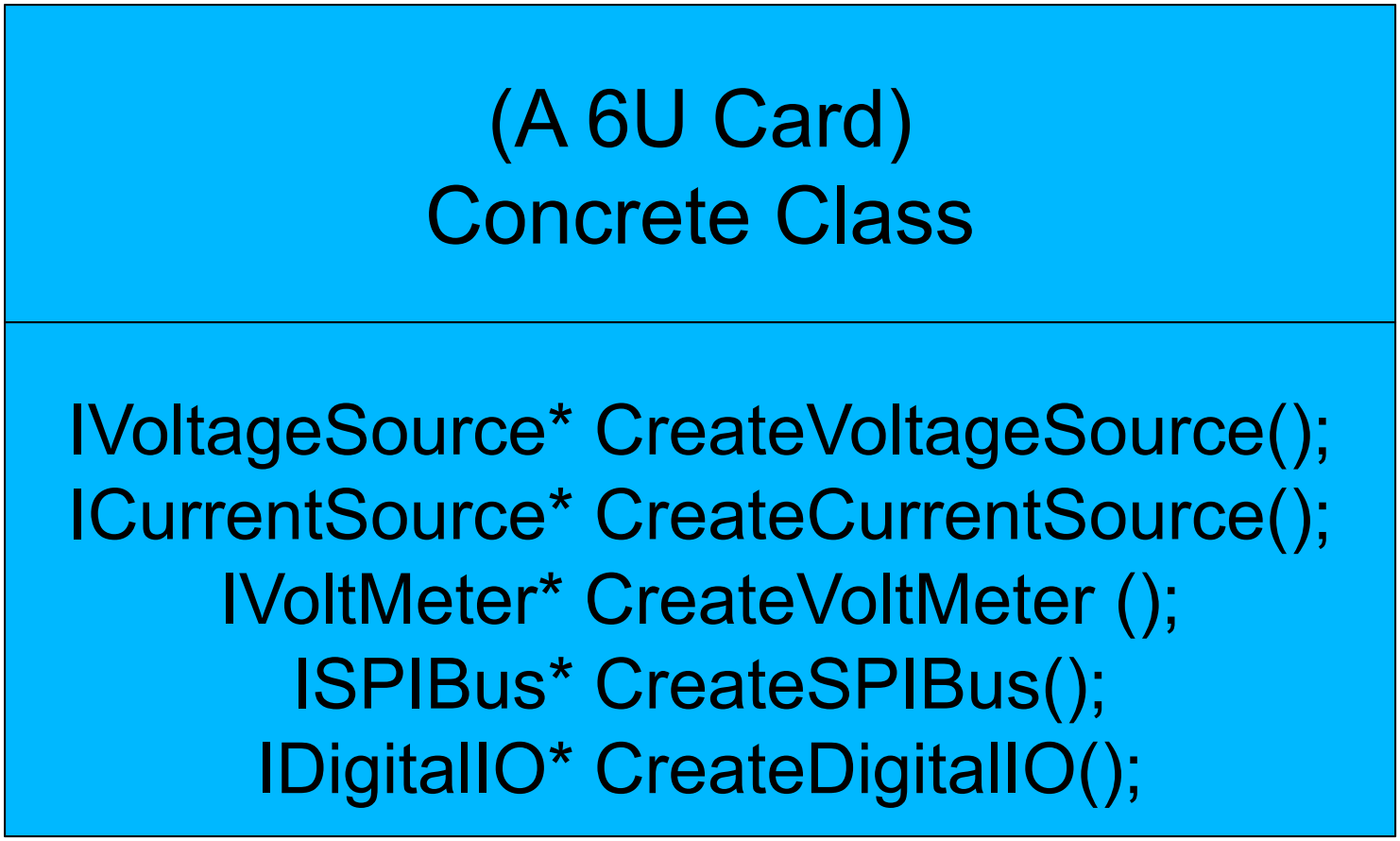
Design Patterns Utilized

- Singleton
- Interfaces
- Object Factories
- Inheritance
- Polymorphism
- Encapsulation

Interfaces and Object Factories



Utilizing Singleton DP



Components of Lean Software [2]

- Eliminate waste
- Amplify learning
- Decide as late as possible
- Deliver as fast as possible
- Empower the team
- Build integrity in
- See the whole

Eliminate waste

- In code writing
  - For example: Inheriting from a base class that implements the communication protocol so as not to duplicate the code on every instrument that utilizes such protocol.
- In learning
  - Don’t waste time learning/remembering how that communication protocol would be / was implemented.
- In decision making
  - Don’t waste time trying to anticipate likely usage, user’s needs, ways to make the functionality intuitive.

Amplify learning

- Layered libraries and cohesive modules enable development teams to work on decoupled tasks.
- Multiple driver modules can be implemented and tested simultaneously and incrementally.
- Driver modules can be provided to the field engineer for instant feedback.
- Short iteration cycles.

Decide as late as possible

- Instruments are sophisticated
- Their functionalities are numerous
- The ways they can be used are unpredictable
- Agile solution provides hooks.
- Additional features must be demanded.

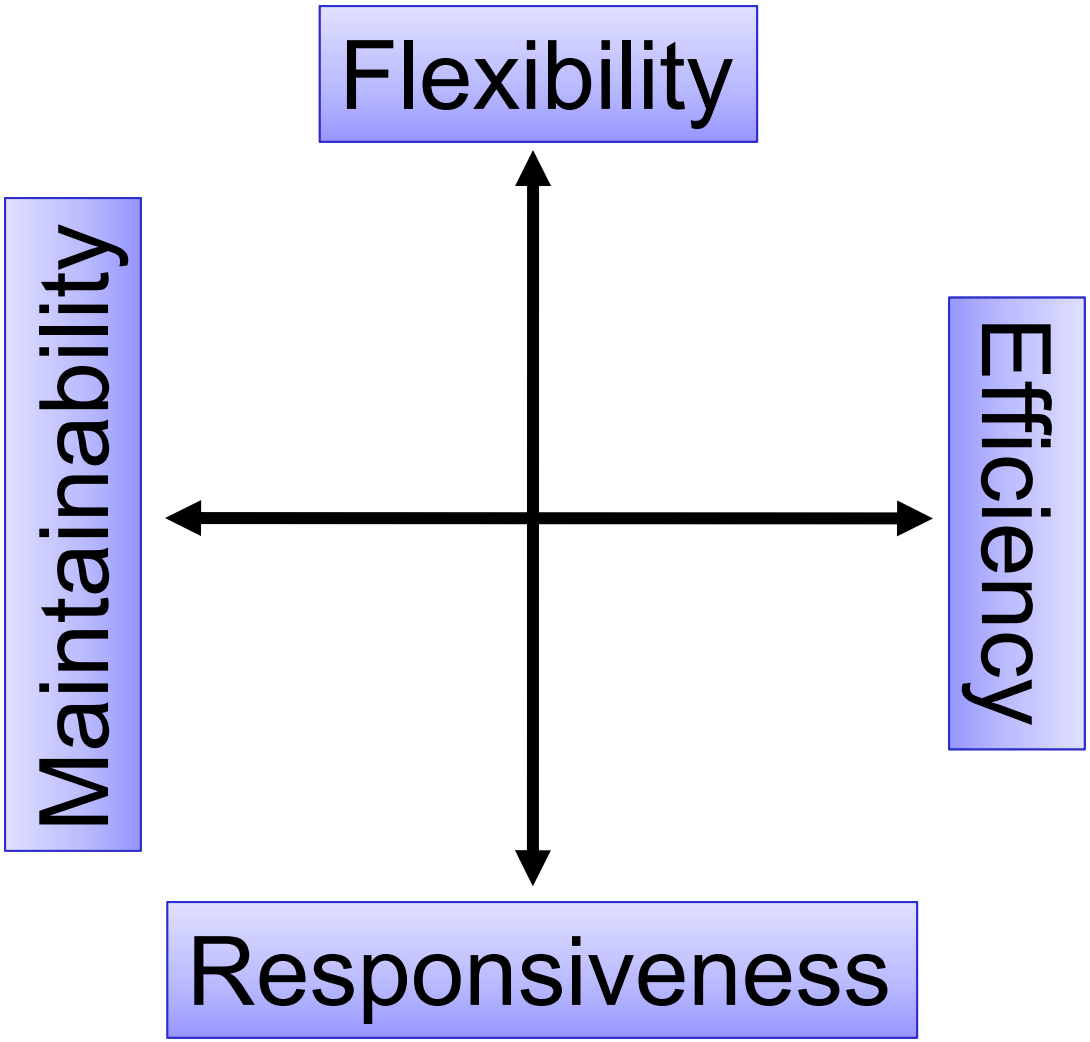
Deliver as fast as possible

- Software is not the ultimate product
- The product is the service the software helps you to provide
- It must not gate the delivery of the service
- To do that, You must deliver what is needed today, not what was needed yesterday or what might be needed tomorrow. Just-in-time!

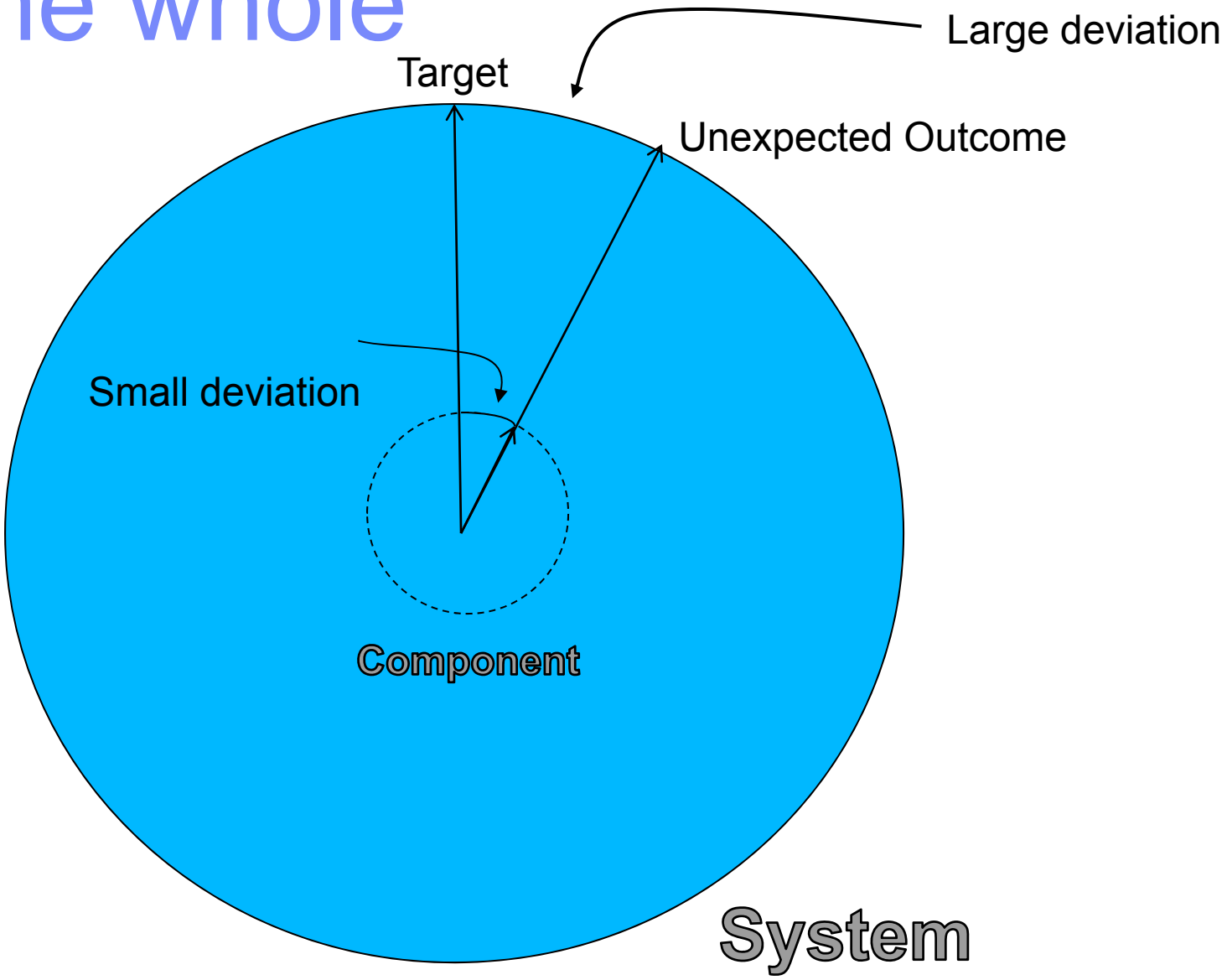
Empower the team

- Encouraging progress
- Catching errors
- Removing impediments
- No micro-managing

Build integrity in



See the whole



Conclusion

- Implemented at IBM RF Test Department
- Agile development
- Short and stable iterations with user feedback at every iteration.
- Alleviates the complexity
- Reduces the learning curve
- Greater flexibility than commercial ATE software

Q&A

THANK YOU

References

- [1] Microwave Journal, March 2013
- [2] Leading Lean Software Development, Mary and Tom Poppendieck, Addison-Wesley, 2010