




# WHAT IS SOFTWARE PERFORMANCE ENGINEERING?

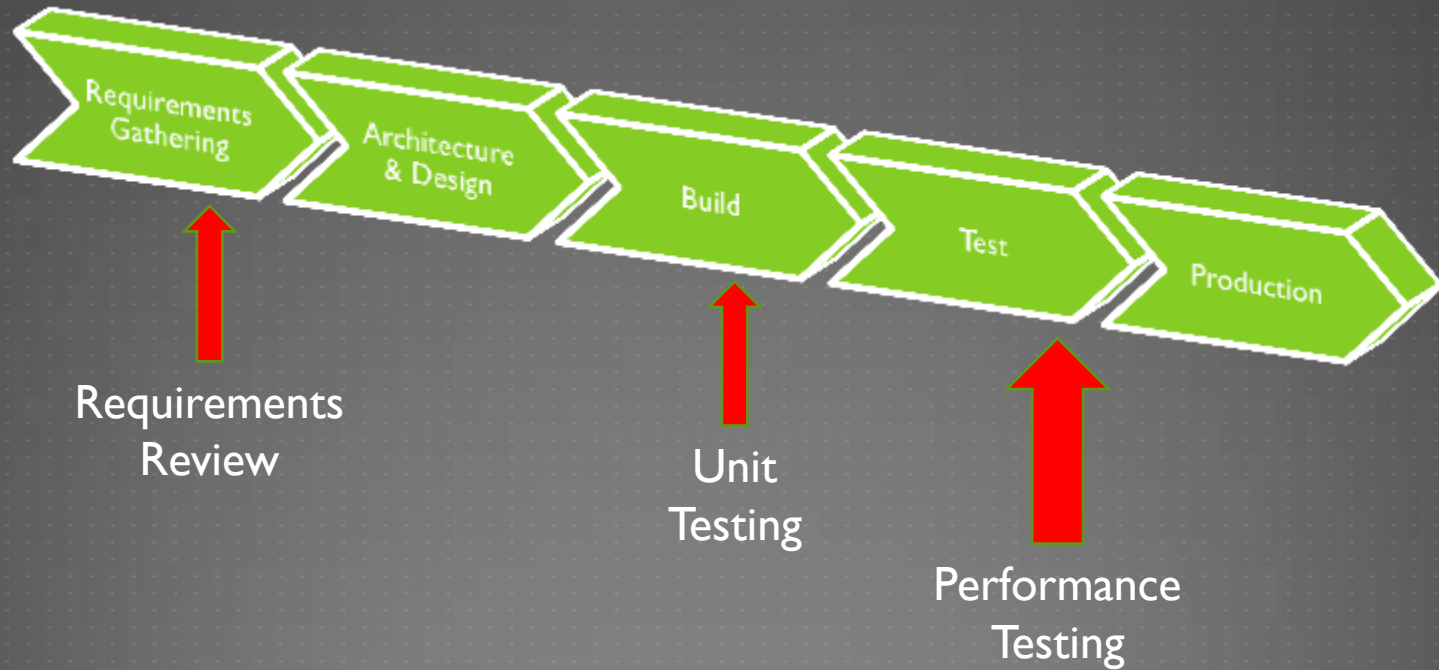
By Michael Foster

[www.cmgaus.org](http://www.cmgaus.org)

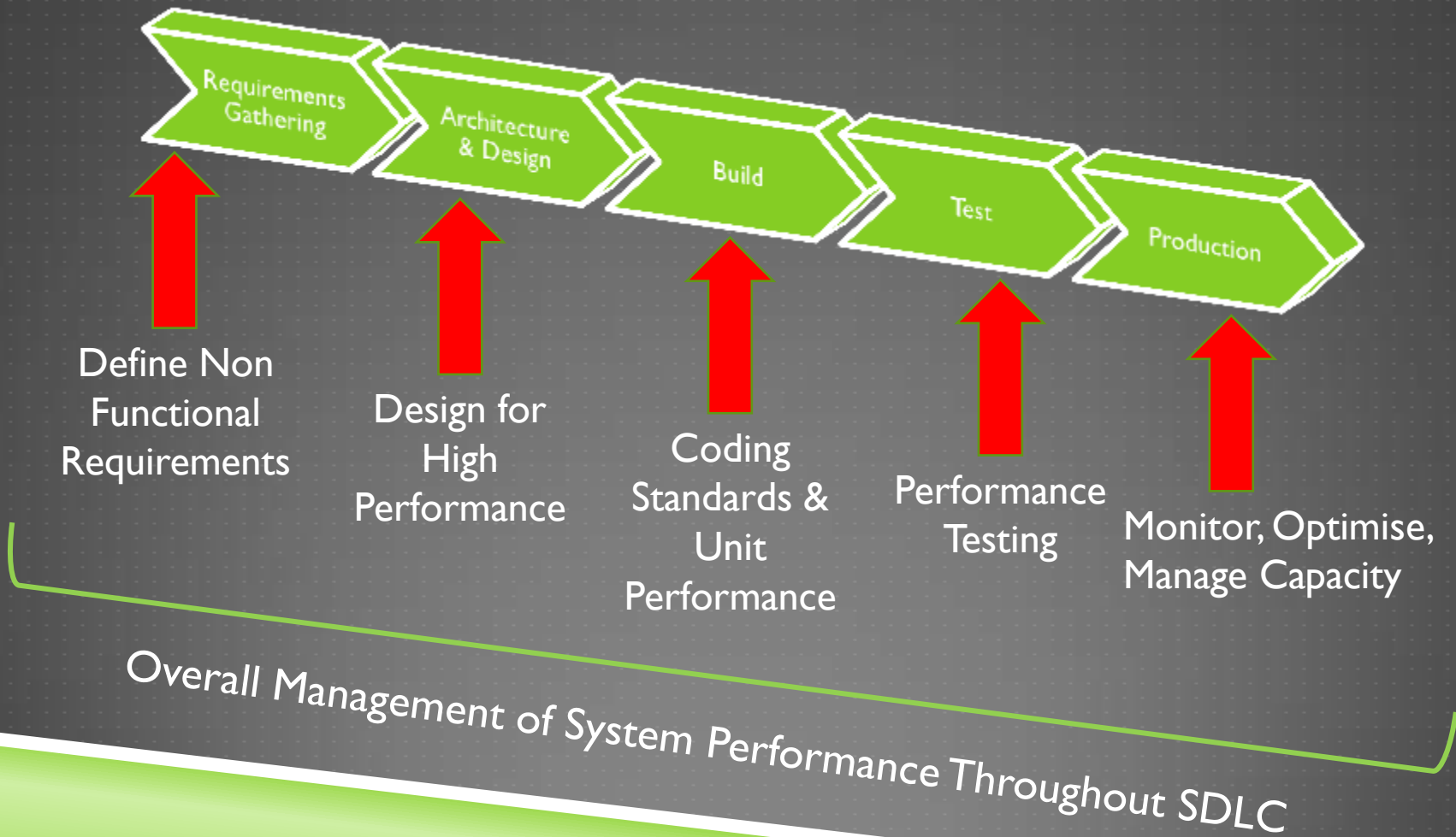
# DEFINITION

- ▶ Software Performance Engineering is:
  - ▶ A systematic and quantitative approach for the cost effective development of software systems to meet stringent Non Functional Requirements.
  - ▶ Or
  - ▶ The set of tasks or activities that need to be performed across the Software Development Life Cycle (SDLC) to meet the documented Non Functional Requirements.
- 

# PERFORMANCE TESTING IN THE SDLC



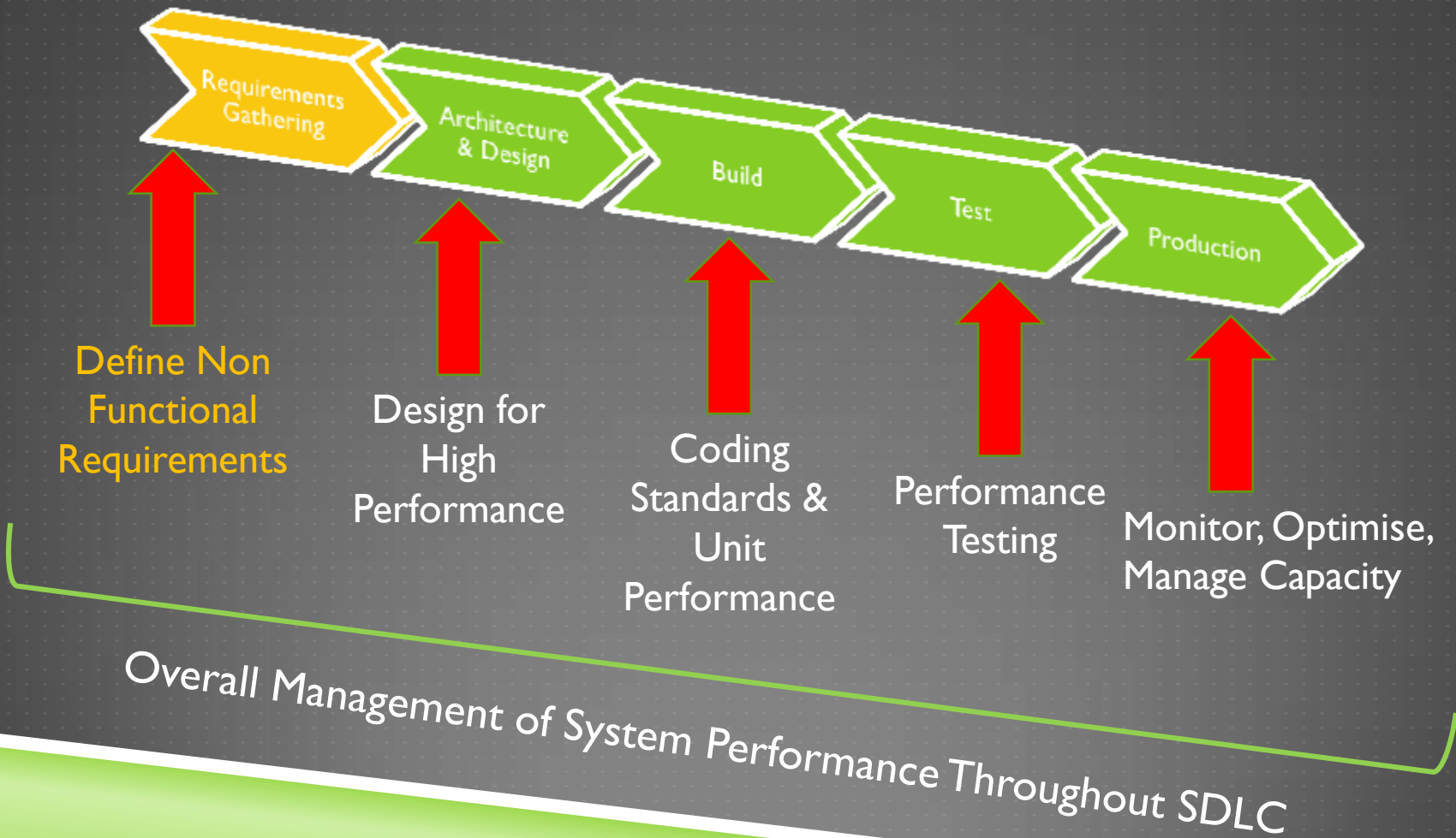
# PERFORMANCE ENGINEERING IN THE SDLC



# ADVANTAGES OF SPE

- ▶ Creating a clear set of Non Functional Requirements lays foundation for successful development.
- ▶ Early and constant focus on system performance at each stage prevents expensive changes late in the project.
- ▶ Performance monitoring in production maintains system performance and reliability, and allows capacity to be expanded before it is exceeded.
- ▶ Proactive approach allows problems to be avoided, keeping focus on the development rather than firefighting.
- ▶ With the successful delivery of a system that performs to the client's requirements, the client gets full value for money.

# REQUIREMENTS ANALYSIS



# DEFINITION – REQUIREMENTS ANALYSIS

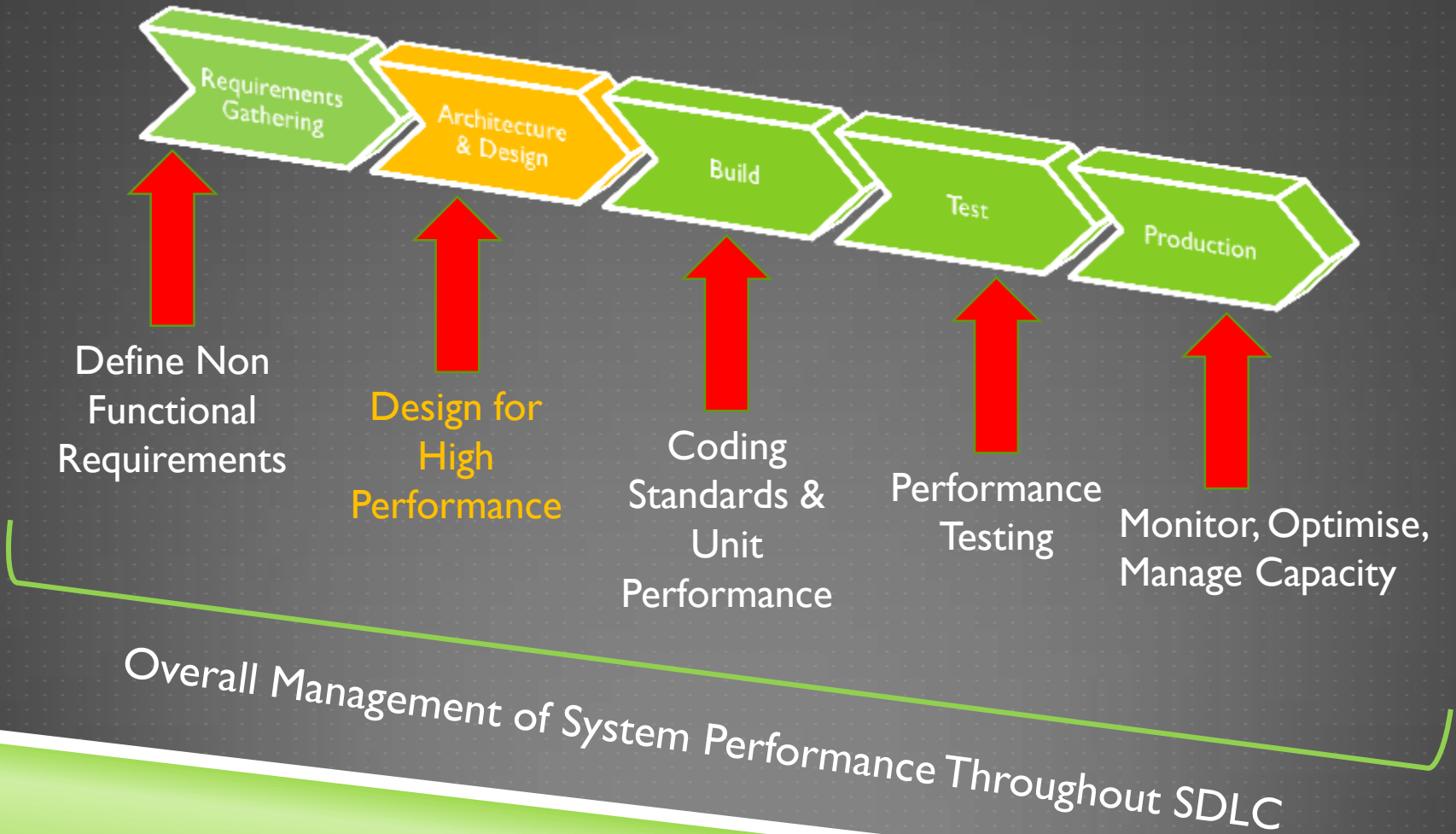
- ▶ Activities performed to identify the Performance Engineering related objectives for the system.
- ▶ Non Functional Requirements include:
  - ▶ Scalability Requirements
  - ▶ Availability Requirements
  - ▶ Reliability Requirements
  - ▶ Performance Requirements
- ▶ These requirements must be documented, socialised and agreed by all stakeholders so that Performance expectations are set early in the development cycle.
- ▶ This phase sets the stage for the rest of the Performance Engineering activities that need to be performed across the SDLC.

# REQUIREMENTS ANALYSIS ACTIVITIES

- ▶ Review Business Requirements and other program documentation – understand the business case and objectives and the platforms being used to deliver them.
- ▶ Review production performance metrics if there is an existing version.
- ▶ Determine Non Functional Requirements – essential so that system performance goals can be set, and measured against.
- ▶ Determine Tools, Resourcing, Infrastructure and Licencing requirements – early identification of these needs allows the program to budget, find resources, purchase tools and hardware, install them and provide training.



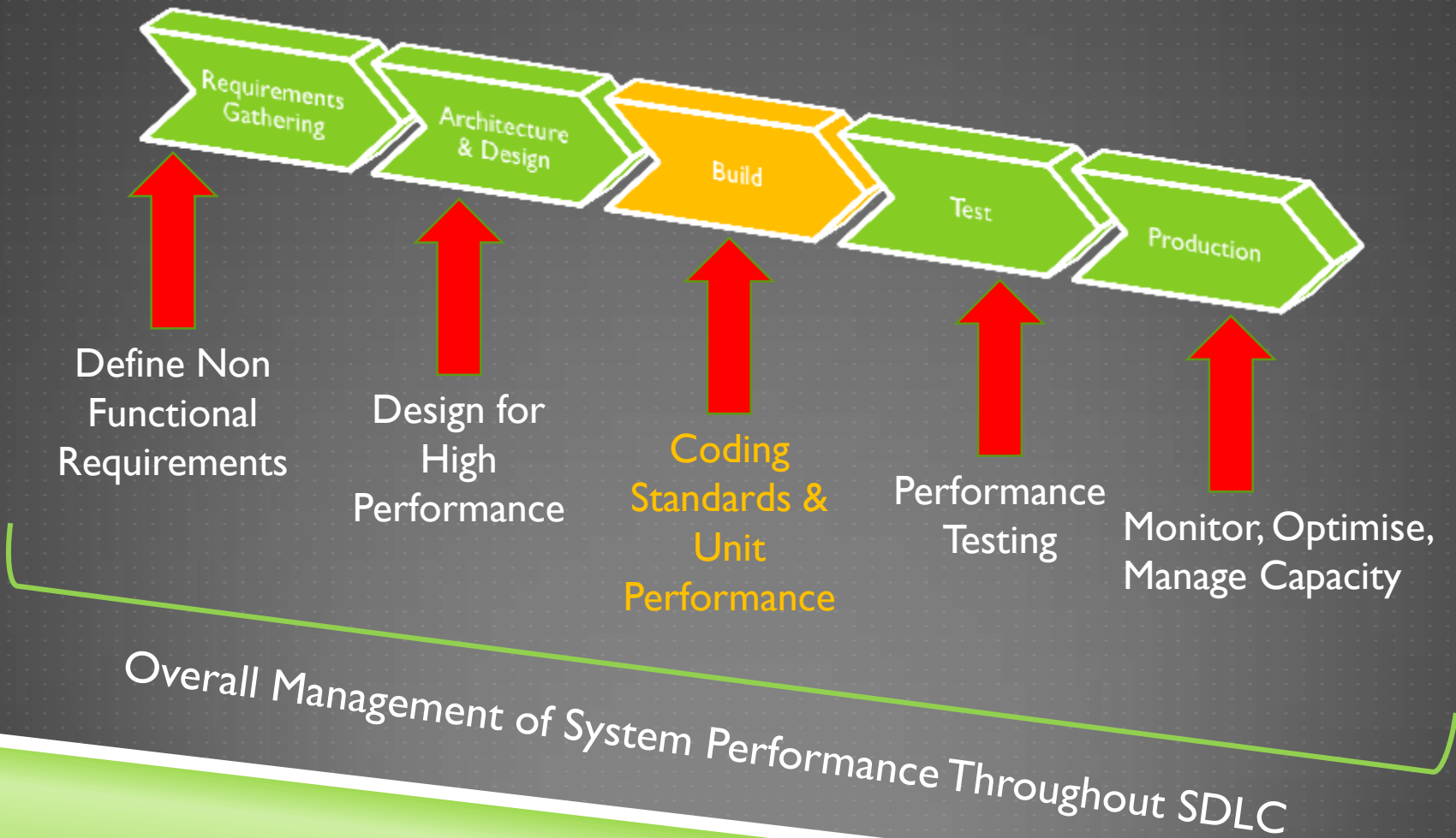
# DESIGN FOR PERFORMANCE



# DESIGN PHASE ACTIVITIES

- ▶ Validate Architectural Options – provide input from a performance perspective to the architecture being recommended.
- ▶ Determine Infrastructure Capacity Required – By combining the Non Functional Requirements with the Architecture design, determine the underlying infrastructure requirements.
- ▶ Set Performance Targets for Developers – Single user performance targets for the development teams across application components and tiers. These are used for unit performance tests.

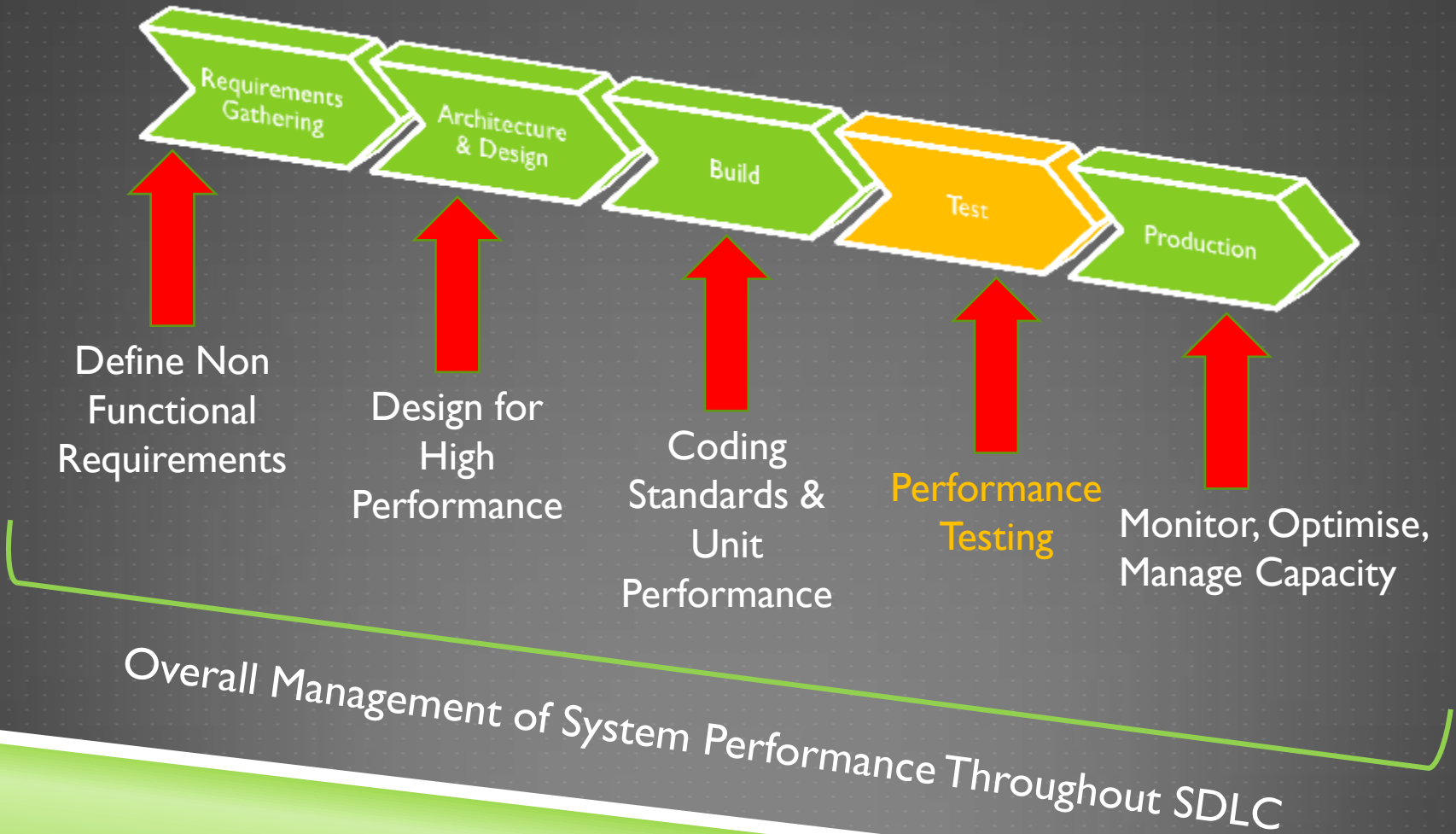
# BUILD FOR PERFORMANCE



# BUILD PHASE ACTIVITIES

- ▶ **Oversee the development and monitor Unit Performance Testing.**
- ▶ **Develop Workload Models:**
  - ▶ **Business Workload** – the set of activities that the users will undertake on the system to achieve business goals. Including any peak load periods or regular cycles (monthly, quarterly), expressed as Transactions per hour.
  - ▶ **Infrastructure Workload** – The workload on the underlying infrastructure CPU, Memory, & Network utilisation etc.
- ▶ **Set up Performance Monitoring** – Install and configure application and infrastructure monitoring tool(s) to measure the application's performance against the SLAs.

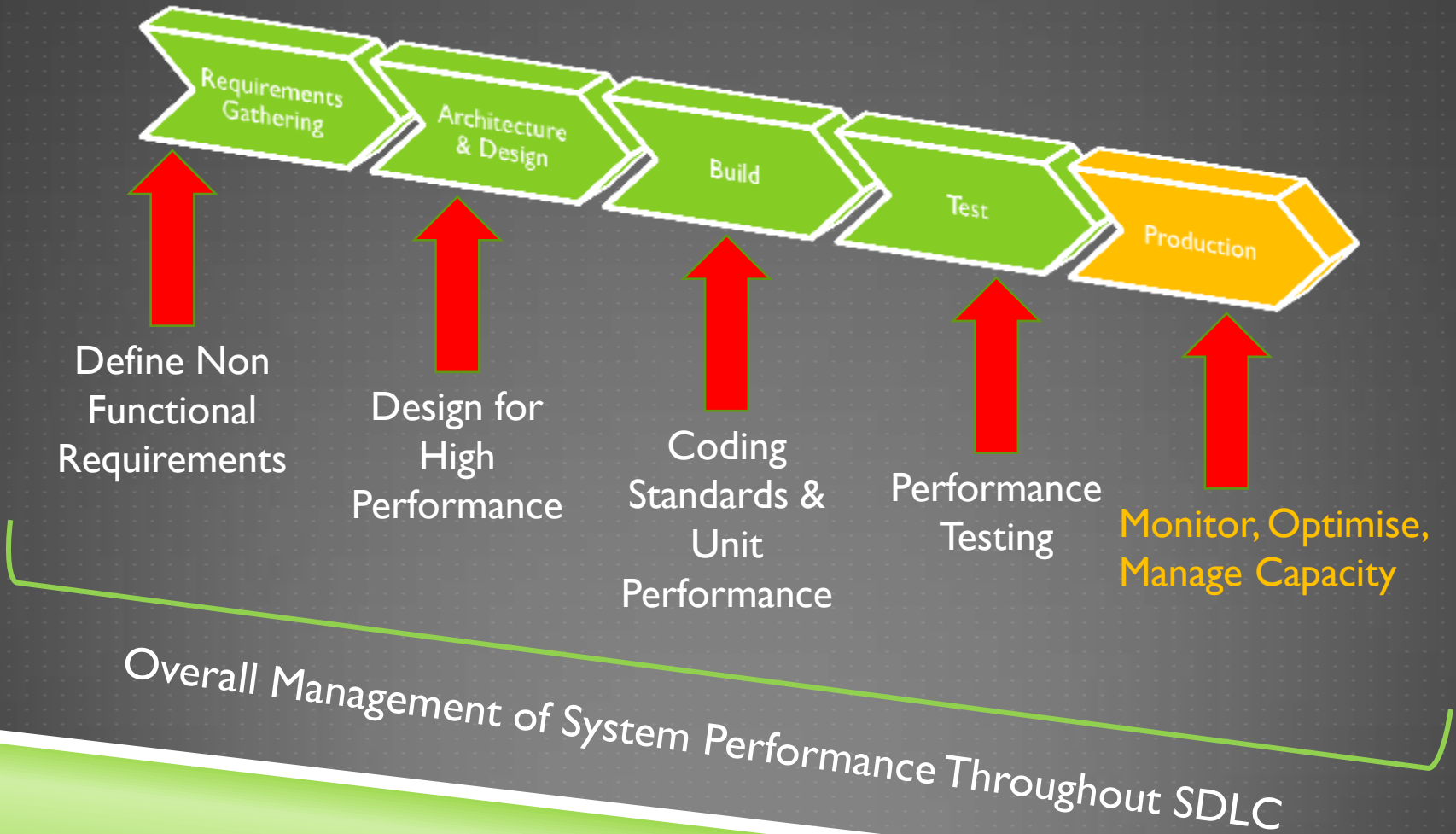
# PERFORMANCE TESTING



# PERFORMANCE TESTING ACTIVITIES

- ▶ Build a set of Performance Tests that will simulate the Workload Model.
- ▶ Use the tests to validate the Non Functional Requirements.
- ▶ Use Performance Monitoring to identify application bottlenecks.
- ▶ Identify application breaking point.
- ▶ Validate the impact of code and configuration changes on application performance.
- ▶ Provide a Pass/Fail result to the program on whether the Non Functional Requirements have been met.

# PRODUCTION



# MAINTAINING PERFORMANCE IN PRODUCTION

- ▶ Use Performance Monitoring to constantly assess application performance, and to identify when the system is approaching its capacity.
- ▶ Use Capacity Management to provide the required infrastructure capacity to sustain growth in business workloads.
- ▶ Provide production workload data to the program that is developing the next release of the application.



# RESOURCES



[www.cmgaus.org](http://www.cmgaus.org)



[www.practicalperformanceanalyst.com](http://www.practicalperformanceanalyst.com)

# QUESTIONS

