SOFTWARE OBSOLESCENCE AND SOFTWARE MAINTENANCE – A TALE OF TWO COST DRIVERS

Dr Sanathanan Rajagopal

Head of Cost Analysis

Company Overview

- Founded Dec 2019
- Consultancy
  - Specialise in delivery of analysis to enable customers to make informed investment decisions & deliver effective projects
- Core skills
  - Operational Research (OR), Cost Analysis, Systems Engineering, Approvals + P3M
- Sectors
  - Defence / Security / Space
  - Energy / Utilities / Nuclear /Transport
- March 2023 – Sirius Digital Founded
  - Software development – Synthetic Environments; AI; ML

Scope of Services

SIRIUS ANALYSIS

P3M

- P3M Design, implementation & Strategic Advice
- P3M Maturity Assessments
- Change Management and transition support
- PMO Improvement initiatives
- End to end project management & project leadership
- Bid Management
- Collaboration leadership

OA / OR

- Soft methods & problem structuring
- Historical trends & data analysis
- Modelling, simulation & wargaming
- R&D, concepting, experimentation
- Capability planning
- Acquisition

Systems Engineering

- Product, service & enterprise level
- Tailorable & scalable lifecycle solutions
- Requirements, Architecting
- Systems Analysis
- Integration Verification Validation & Testing
- Sustainment
- Disposal & Replacement

Cost Analysis

- Whole life cost models
  - Should-/Could-cost
  - Cost-Benefit
  - Value-For-Money
- Cost estimation
  - Acquisition
  - Through-life
  - All domains, platforms & DLODS
- Investment Appraisals
- Risk Analysis
- Programme Mgt
- Business Cases

Approvals

- Business Case support
- 5 Case Model
- HMT Approvals
- Cost-benefit, cost-effectiveness and Option selection
- Value for Money cases
- Financial estimation
- Risk management
- Socio-economic benefits

© Sirius Analysis Limited

Customers & Partners

Ministry of Defence
AWE
ALLIED COMMAND TRANSFORMATION
SVGC
BAE SYSTEMS
CORDA
ARMED FORCES COVENANT
PA
Department for Business, Energy & Industrial Strategy
Human Social Science Research Capability
Ofwat
THALES
CERVUS
UNIVERSITY OF PORTSMOUTH
Futures Lab
Powered by EDP
tpgroup
AURORA
ENGINEERING PARTNERSHIP
QinetiQ | Atkins | BMT
UK SPACE AGENCY
ASTRID
ANALYSIS FOR SCIENCE & TECHNOLOGY RESEARCH IN DEFENCE
UK Atomic Energy Authority
Home Office
FRAZER-NASH CONSULTANCY
RAND CORPORATION
ROYAL NAVY
Arke Ltd
[dstl]
The Science Inside
Squarcle
QINETIQ
ofgem
Making a positive difference for energy consumers
ORR
Serapis Framework

Agenda

- Introduction
- Software Obsolescence vs Software Maintenance
- Software Obsolescence
- Software Obsolescence Key Cost Drivers
- Software Maintenance
- Software Maintenance Key Cost Drivers
- Conclusion – why estimating this is important
- Discussions

Introduction – Software Maintenance

Software Maintenance is defined as “ the process of modifying a software systems or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment”

-IEEE, 1990

“ Software Maintenance is the totality of activities required to provide cost-effective support to a software system. Activities are performed during the pre-delivery stage as well as the post-delivery stage. Pre-delivery activities include planning for post-delivery operations, supportability, and logistics determination. Post-delivery activities includes software modification, training and operating a help desk”

-Thomas Pigoski, “Practical Software Maintenance – Best practice for managing your software Investment”

Types of Software Maintenance

Perfective Maintenance	Preventative Maintenance	Corrective Maintenance	Adaptive Maintenance
Perfective maintenance is the modification of a software application, after delivery, to improve performance or maintainability	The modification of a software application after delivery to detect and correct latent faults in the software product before they become effective faults	The reactive modification of a software product performed after delivery to correct discovered problems.	Enhancements necessary to accommodate changes in the environment in which a software product must operate

Why is Software Maintenance required

The Software Maintenance is required in the system to satisfy the user for the system upkeep requirements.

Some of these requirements may include the below -

- Bug fixes
- Address fault/failures, security patches etc.
- Review of the stored files to ensure they are still useable
- Upgrade the software to enhance its capability
- Taking care of the current versions to ensure that its up and running and meeting the requirements.

Different activities under software maintenance

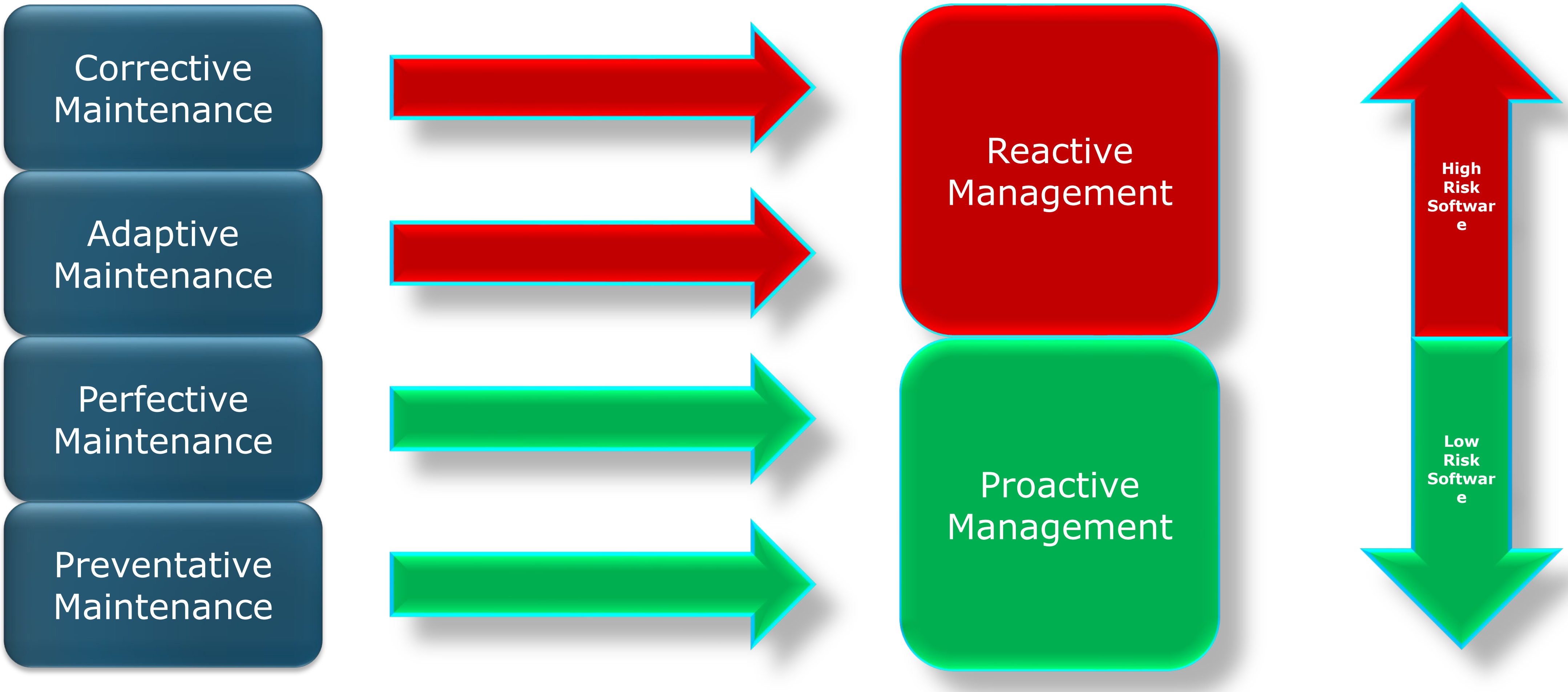
ISO/IEC 12207 identifies the primary activities of software maintenance as:

- Process implementation
- Problem and modification analysis;
- Modification implementation
- Maintenance review/acceptance
- Migration
- Software Retirement.

Software Maintenance has huge costs involved

- As per various researches done on Software Maintenance, it is observed that it accounts for almost 60-80% of the overall build costs.
- Maintenance costs are primarily attributed to enhancements than corrections to the applications or systems involved.
- To determine the impact of changing the system on rest of the application portfolio requires in depth analysis and retesting of the system. This adds up to the major challenges faced to deliver the change requirement.
- Team should have experienced staffing and knowledge of the system to keep the costs low.

Recommended Software Maintenance Strategy



QINETIQ/EMEA/CIT/CP1901699| ©QinetiQ

Estimating Maintenance- Traditional

Based on the type of activity required for maintenance, the method followed may be different. These may include the below

1. Estimation by analogy
2. Bottom's up estimation
3. Function point sizing for sizing minor and major enhancements
4. Ticket based estimation
5. Other Proprietary Methods

To get to the overall estimate, enumerate all the various types of work item in scope and then use the related data to arrive at the effort estimates for each work type. Remember not all methods are applicable for all work types.

Estimating Maintenance- Other Proprietary Methods

- Organisation may derive their own estimation method based on their historical data and scope of work they support.

  For example, an organization may assign a weighting factor to complexity of each work type and then assess the total work effort based on number of work types and their complexity.

- Large historical data is required to validate such data models which they may need to be recalibrated with time

Other Metrics used for maintenance projects

- Maintenance Productivity: Number of application functions points maintained per 1000 hours of labor
- Defect Severity : It is the number of valid defects that occurred in support or production for the portfolio during a 12-month analysis time frame. Defect severity can be classified into different subcategories like Critical, Major, Minor , Cosmetic.
- Number of Defects per thousands of function points, post implementation provides insight into the quality and reliability of currently running applications
- Other Metrics include
  - Defect Removal efficiency
  - Percentage SLA adherence
  - Response Time
  - Percentage Availability

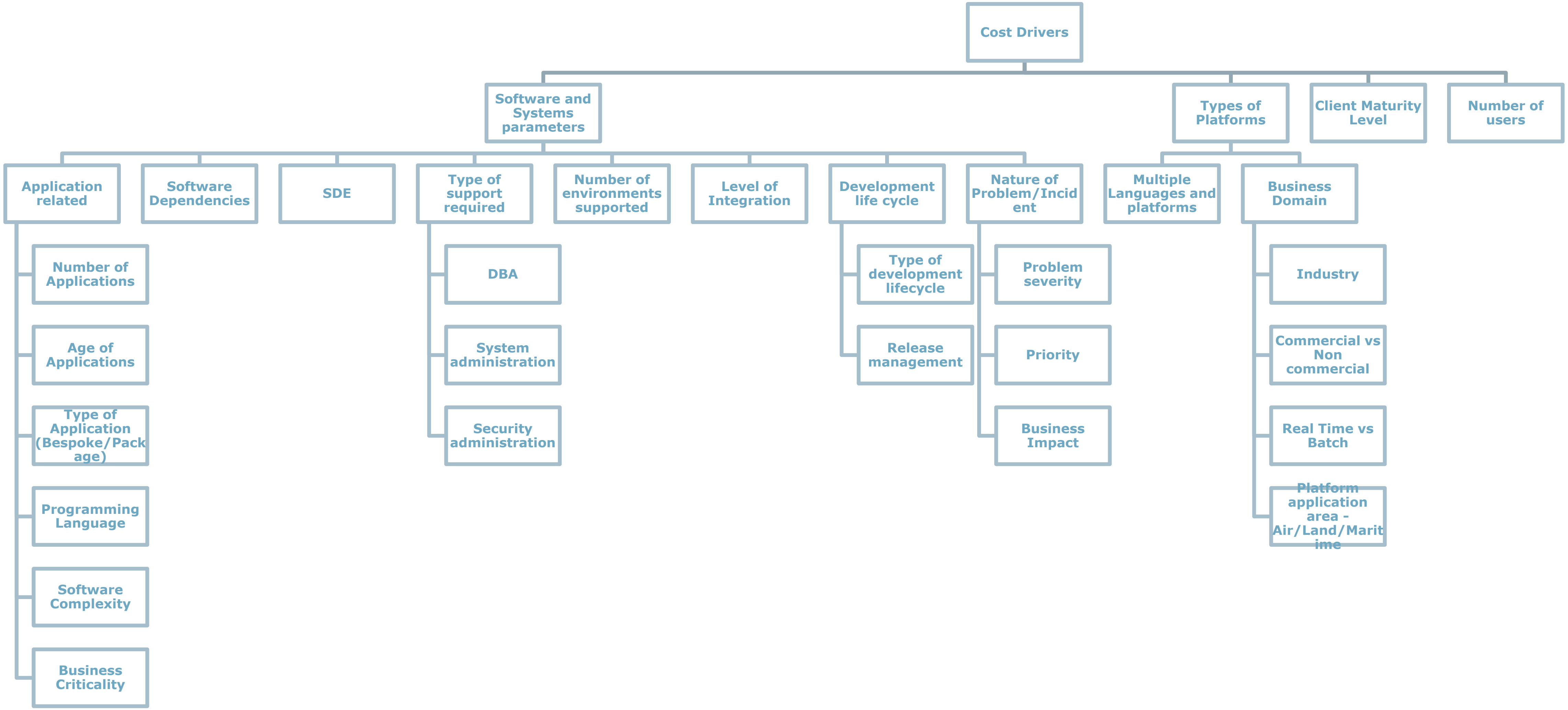
Some Rules of Thumb

Below are some rules of thumb for Maintenance work by Capers Jones using Function points as sizing method

- Sizing Software Defect Potentials:
  - Function points raised to the 1.25 power predict the approximate defect potential for new software projects.

- Estimating Software Maintenance Staffing Levels:
  - Function points divided by 750 predict the approximate number of maintenance personnel required to keep the application updated.

Cost Drivers to consider when estimating Software Maintenance Cost



- Cost Drivers
  - Software and Systems parameters
    - Application related
      - Number of Applications
      - Age of Applications
      - Type of Application (Bespoke/Package)
      - Programming Language
      - Software Complexity
      - Business Criticality
    - Software Dependencies
    - SDE
    - Type of support required
      - DBA
      - System administration
      - Security administration
    - Number of environments supported
    - Level of Integration
    - Development life cycle
      - Type of development lifecycle
      - Release management
    - Nature of Problem/Incident
      - Problem severity
      - Priority
      - Business Impact
  - Types of Platforms
    - Multiple Languages and platforms
    - Business Domain
      - Industry
      - Commercial vs Non commercial
      - Real Time vs Batch
      - Platform application area - Air/Land/Maritime
  - Client Maturity Level
  - Number of users

Software Obsolescence

Software Obsolescence is defined as " what happens when the original and authorised third party ceases to provide support with regular update, upgrade, fixes or due to the changes in target or operating environment, systems or hardware which makes the software unusable"

-S Rajagopal et al; (2014)

What is Obsolescence ?

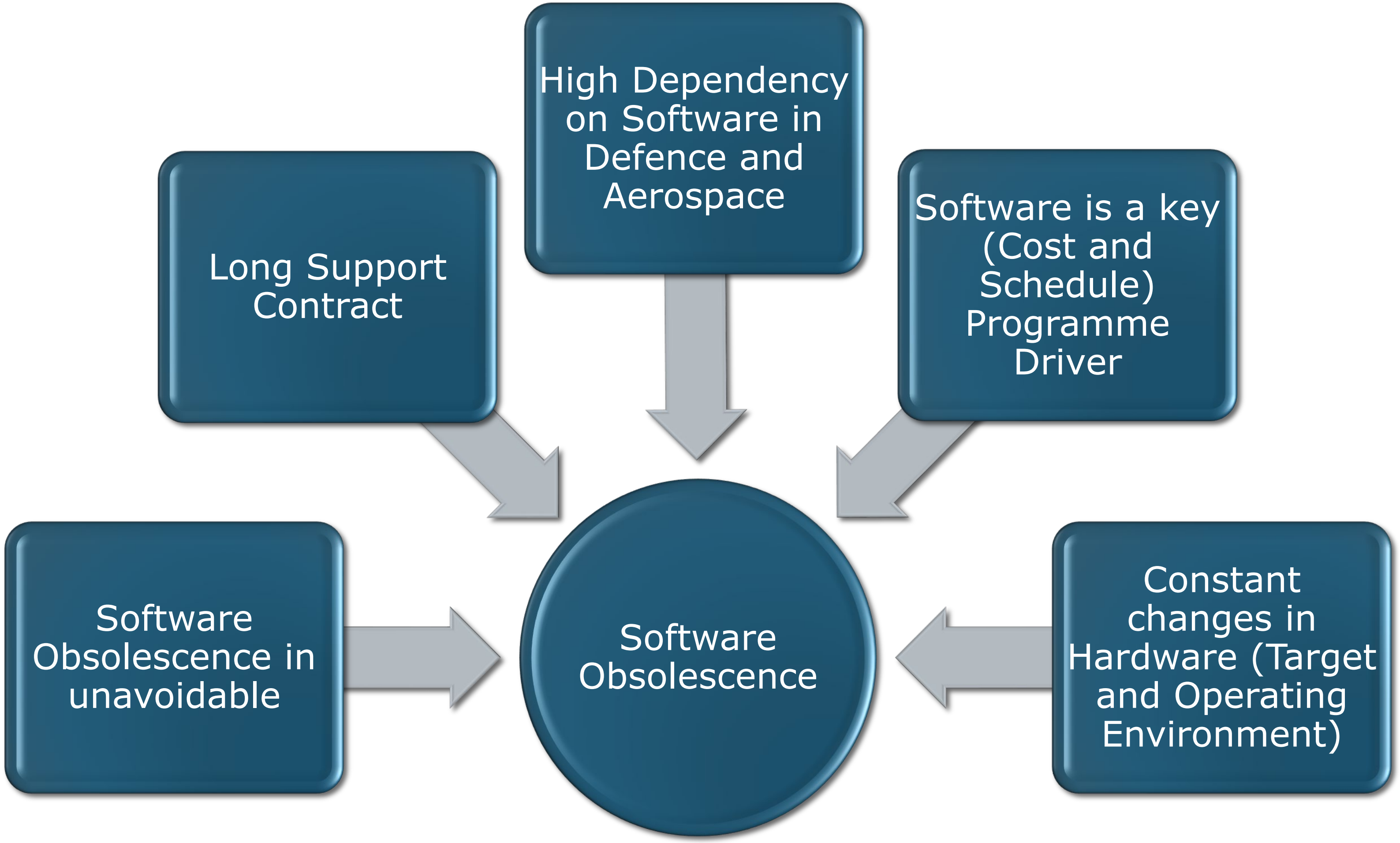
- There are various definitions for Obsolescence in use however IEC 62402:20071 defines obsolescence as

"Transition from availability from the original manufacturer to unavailability"

- Obsolescence Management is the

"The coordinated activities to direct and control an organisations with regard to Obsolescence"

Why is Software Obsolescence important ?



High Dependency on Software in Defence and Aerospace

Long Support Contract

Software is a key (Cost and Schedule) Programme Driver

Software Obsolescence in unavoidable

Software Obsolescence

Constant changes in Hardware (Target and Operating Environment)

Obsolescence – The Threat

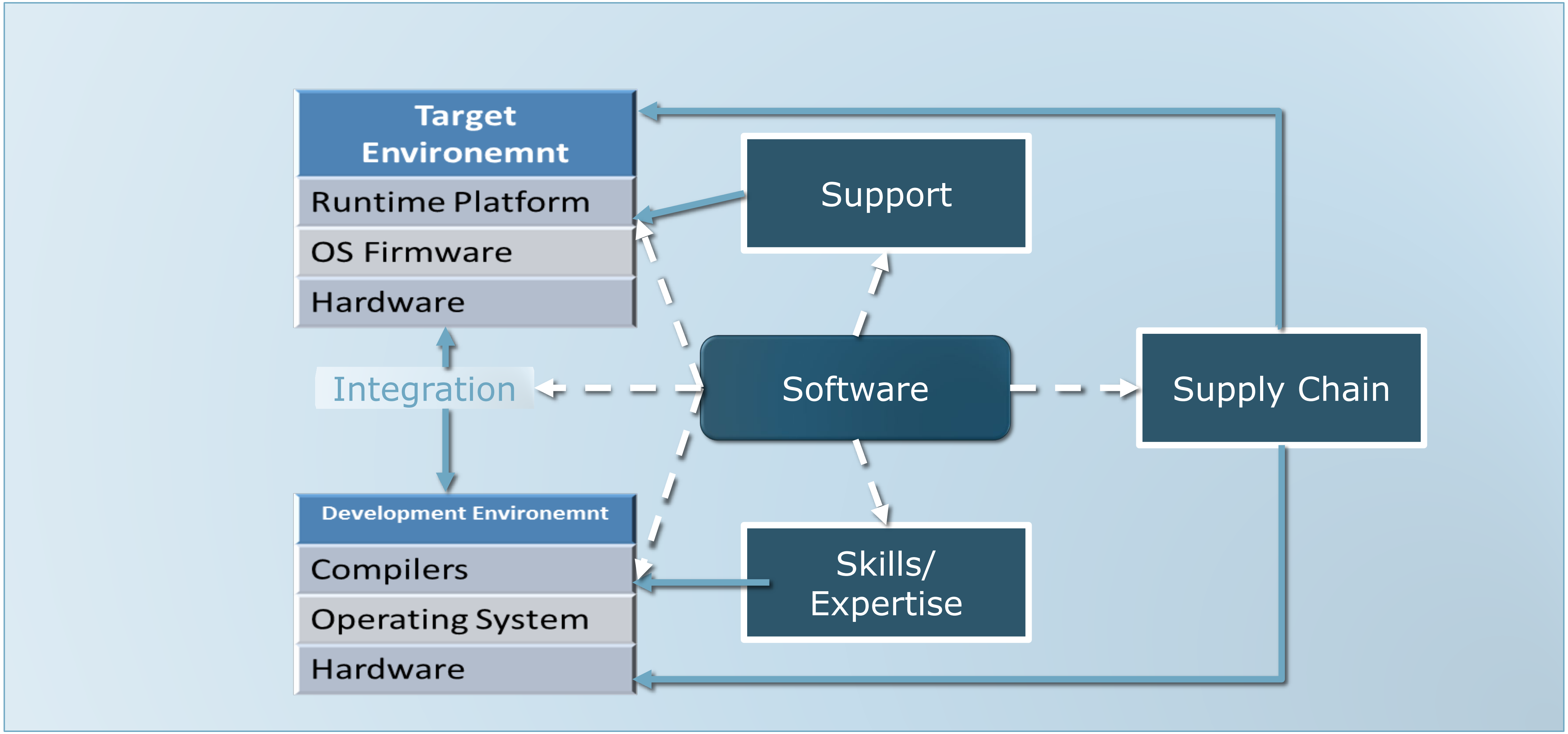
Quotes from the industry to help understand the context

“Obsolescence is bigger issue than reliability”- British Airways Engineering

“Obsolescence is now one of the biggest threats to the future of many sectors of industry” –BAE Systems

“Obsolescecne is number 2 risks to the project” –Eurofighter Typhon

Software Obsolescence Management



Why Software Obsolescence is different to Hardware Obsolescence

- Software defects are design defects
- Software does not have energy related wear-out. Errors can occur without warning
- Software reliability is not a fuction of time
- Environmental factors does not effect software reliability
- Software reliability cannot be predicted from any physical basis, it depends completely on human factors in design
- Software interface are purely conceptual othe than visual

Software Maintenance Vs Software Obsolescence

Software Maintenance	Software Obsolescence
Bug fixes	Replacement of entire application if need be to a new one
To address fault/Failures, security patches etc.	To address the issues with the application in totality
Maintenance is the review of the stored files to ensure they are still useable	Solves unavailability of fixes, licenses, permission and upgrades
Software maintenance takes care of the current versions to ensure that its up and running and meeting the requirements	Software Obsolescence management looks forward the industry standards and other software to continue supportability of the software
Maintenance deals with the upgrading the software to enhance capability	Obsolescence management deals with enforced changes in the environment

Types of Software Obsolescence

- Software gets obsolete due to one of the following reasons (P Sandborn, et al)
  - **Functional obsolescence**: *If there are changes to the hardware, system or other software in the same system.*
  - **Technological Obsolescence**: *his happens when vendor stops supporting the products or unavailability of the software in market etc.*
  - **Logistical Obsolescence**: *This happens when the media or the hard drive for example does not support the software*.

What Triggers Software Obsolescence

- Applications not supported by the developers
- Changes in
  - development environment,
  - Integration and test environment,
  - target environment and adjacent systems
- Compatibility issues, both backward and foreword
- Technology insertions
- Hardware obsolescence
- Changes in safety and legal requirements

Software Obsolescence Management

- There are mainly two types of Obsolescence Management Strategy
  - Proactive
  - Reactive
- These depends on various factors such as, but not limited to
  - Resource availibilty
  - Obsolescence management
  - Obsolescence monetering
  - Supply chain moneteering

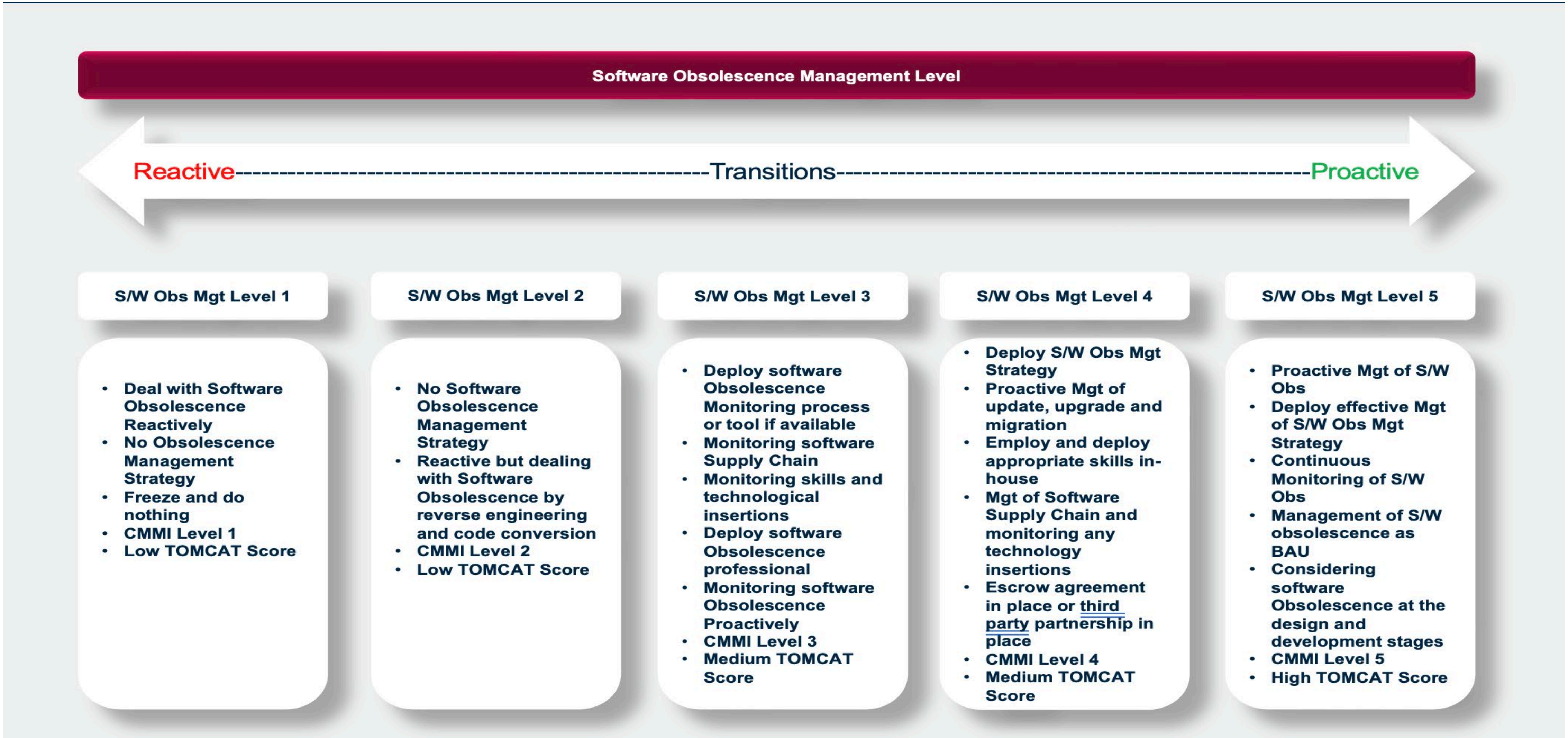
Mitigating Software Obsolescence

- Maintaining Software Obsolescence Register and constantly monitoring
- Through development of appropriate skills
- Realizations that Software Obsolescence will happen and cannot be dealt with as an afterthought, it has to be considered up-front frim day one.
- Contractual agreement with OEM to keep the supportability as long as required.
- Proactive strategies to identify the software and hardware to ensure that both are available or resolve prior to an arising
- Escrow methodology
- Support plaining

Proactive Software Obsolescence Management

- A true forecast of industries plans for software sunset and new developments
- An appropriate contractual model involving the customer, contractor, supply chain that recognize the costs centers for the facilities and capabilities needed, and the allocation of sufficient budget for both proactive and reactive elements.
- Understanding of the use of the software in the hardware and the restraints in its use and replacement
- Financial support
- SQEP
- To have full understanding of system architecture to be able to target key items for monitoring
- Rapid availability of cost and schedule revisions to test alternative mitigation strategies

Key Cost Drivers – Software Obsolescence Management



Software Obsolescence Management Level

Reactive--Transitions--Proactive

S/W Obs Mgt Level 1

- Deal with Software Obsolescence Reactively
- No Obsolescence Management Strategy
- Freeze and do nothing
- CMMI Level 1
- Low TOMCAT Score

S/W Obs Mgt Level 2

- No Software Obsolescence Management Strategy
- Reactive but dealing with Software Obsolescence by reverse engineering and code conversion
- CMMI Level 2
- Low TOMCAT Score

S/W Obs Mgt Level 3

- Deploy software Obsolescence Monitoring process or tool if available
- Monitoring software Supply Chain
- Monitoring skills and technological insertions
- Deploy software Obsolescence professional
- Monitoring software Obsolescence Proactively
- CMMI Level 3
- Medium TOMCAT Score

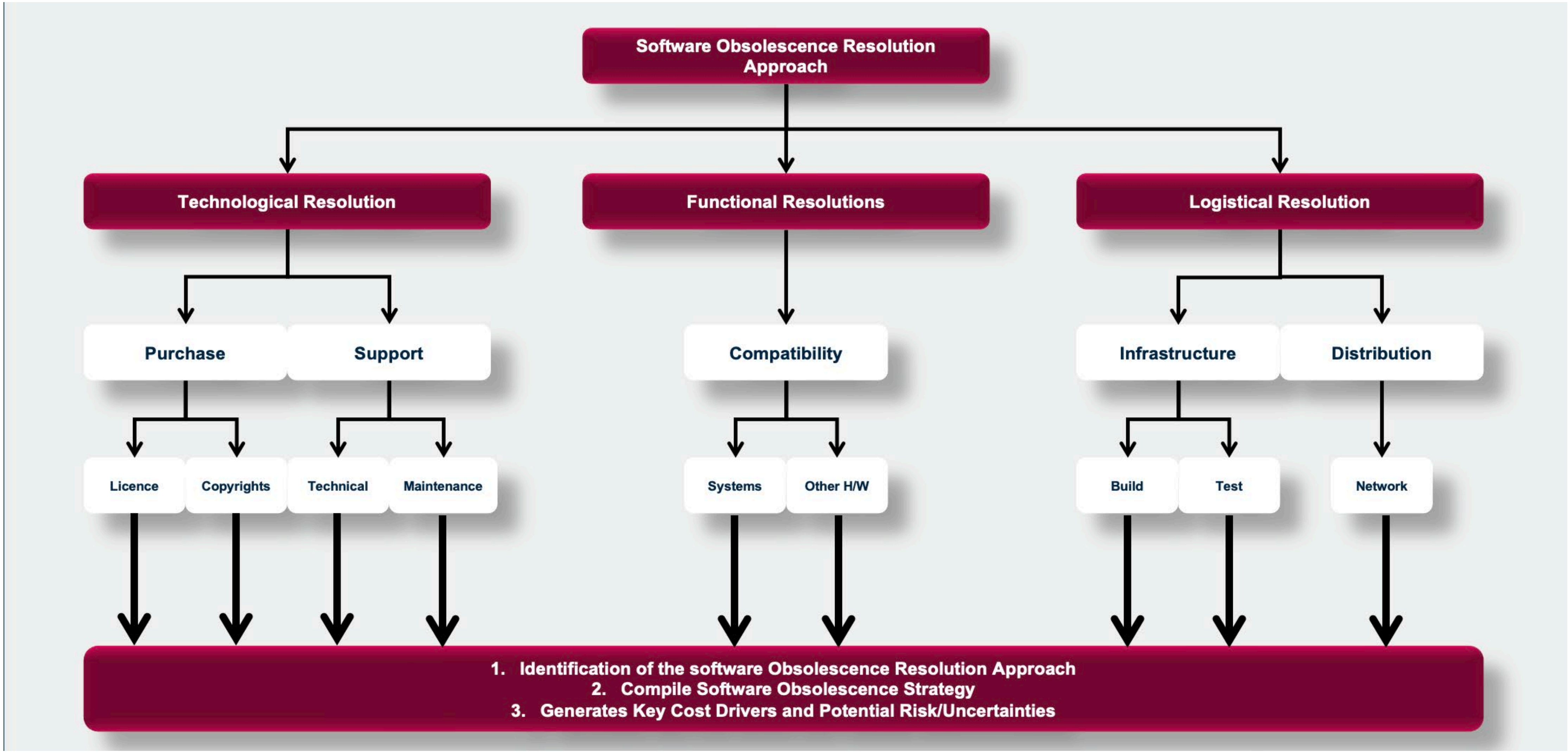
S/W Obs Mgt Level 4

- Deploy S/W Obs Mgt Strategy
- Proactive Mgt of update, upgrade and migration
- Employ and deploy appropriate skills in-house
- Mgt of Software Supply Chain and monitoring any technology insertions
- Escrow agreement in place or third party partnership in place
- CMMI Level 4
- Medium TOMCAT Score

S/W Obs Mgt Level 5

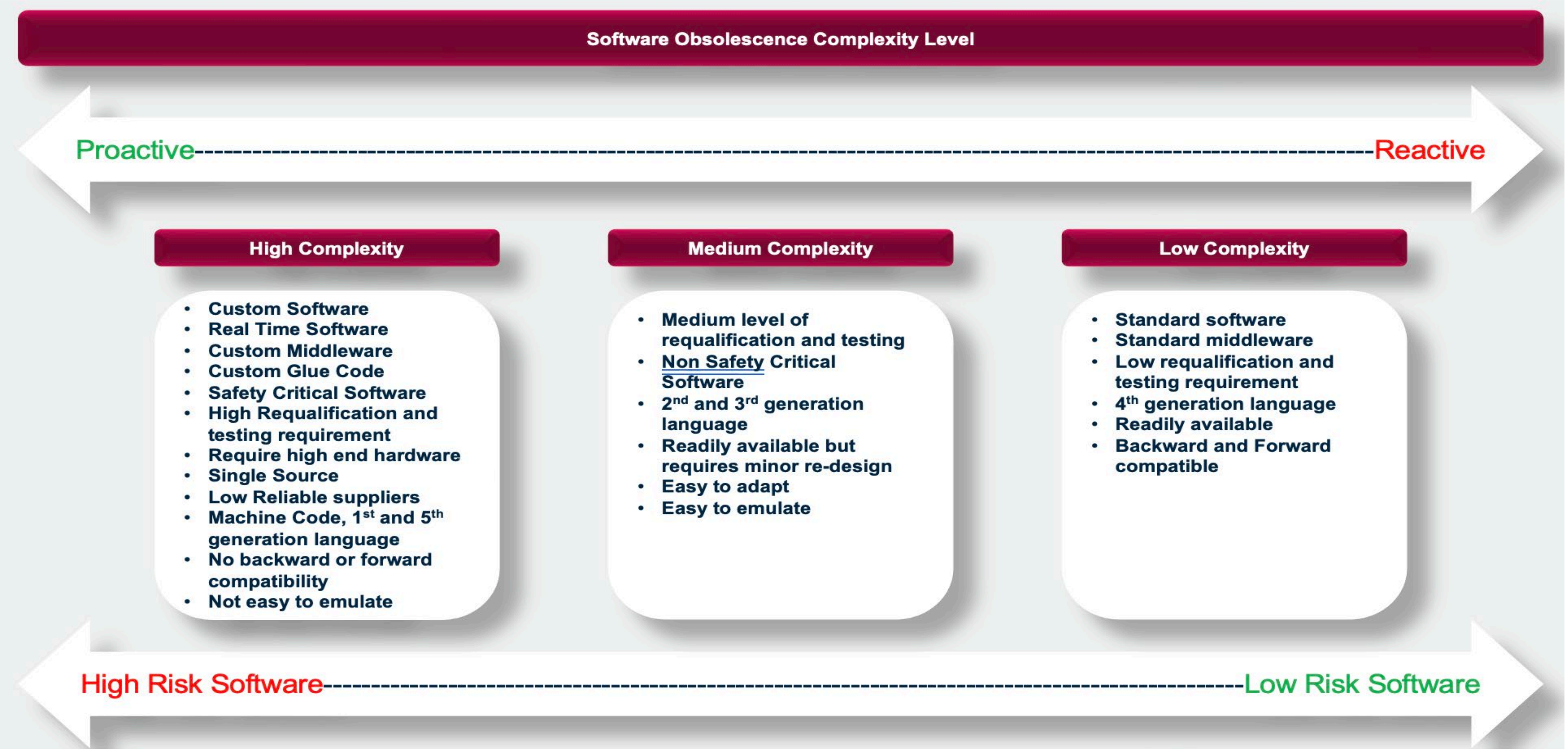
- Proactive Mgt of S/W Obs
- Deploy effective Mgt of S/W Obs Mgt Strategy
- Continuous Monitoring of S/W Obs
- Management of S/W obsolescence as BAU
- Considering software Obsolescence at the design and development stages
- CMMI Level 5
- High TOMCAT Score

Key Cost Drivers – Software Obsolescence Resolutions



1. Identification of the software Obsolescence Resolution Approach
2. Compile Software Obsolescence Strategy
3. Generates Key Cost Drivers and Potential Risk/Uncertainties

Key Cost Drivers – Software Obsolescence Complexity



Software Obsolescence Complexity Level

Proactive---Reactive

High Complexity

- Custom Software
- Real Time Software
- Custom Middleware
- Custom Glue Code
- Safety Critical Software
- High Requalification and testing requirement
- Require high end hardware
- Single Source
- Low Reliable suppliers
- Machine Code, 1st and 5th generation language
- No backward or forward compatibility
- Not easy to emulate

Medium Complexity

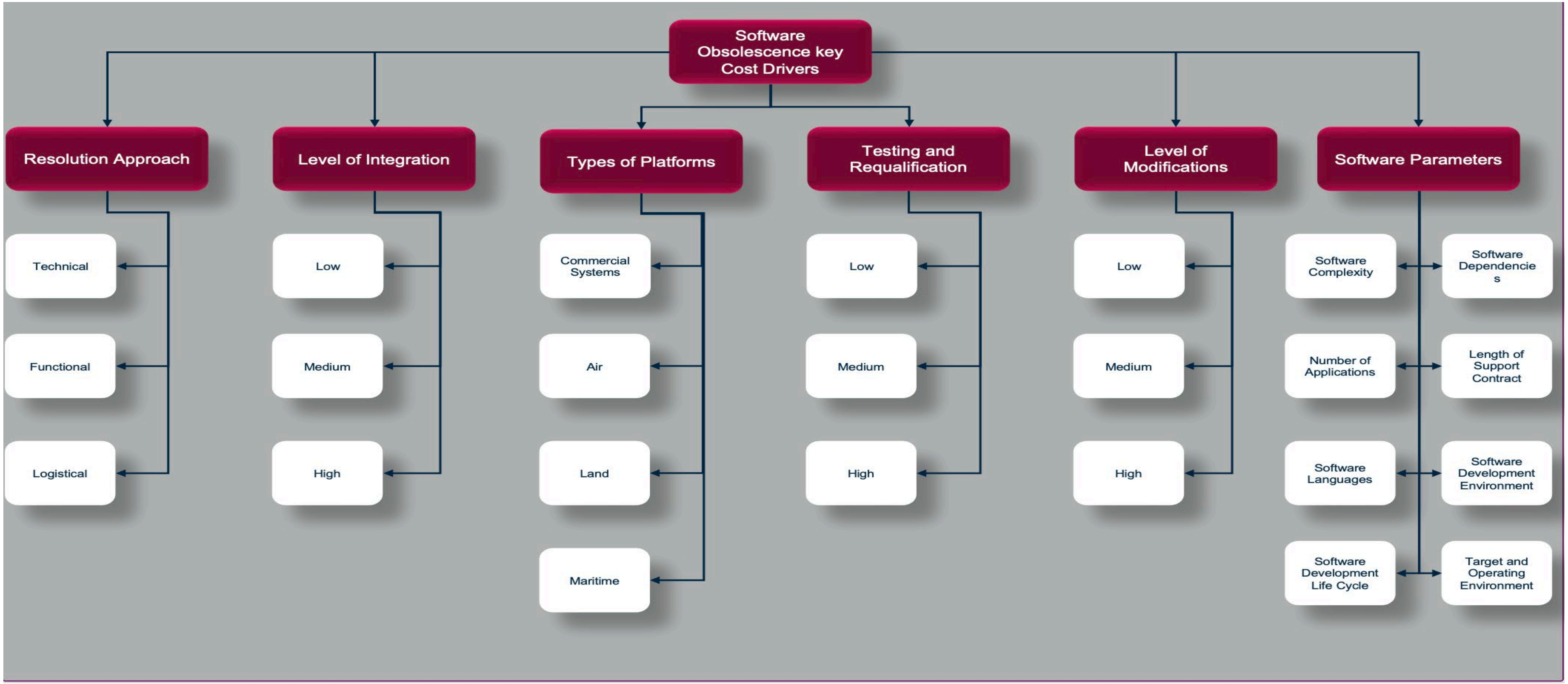
- Medium level of requalification and testing
- Non Safety Critical Software
- 2nd and 3rd generation language
- Readily available but requires minor re-design
- Easy to adapt
- Easy to emulate

Low Complexity

- Standard software
- Standard middleware
- Low requalification and testing requirement
- 4th generation language
- Readily available
- Backward and Forward compatible

High Risk Software---Low Risk Software

Other Key Cost Drivers



Conclusion – why this is important

- We are getting better at estimating the cost of software development
- Defence software especially the bespoke software have very long-shell life.
- Long predictive life cycle of bespoke software triggers constant maintenance and obsolescence and needs to be estimated in order to reduce/manage the through life cost
- Having a good understanding of the key cost drivers of software obsolescenc and maintenance is more robust than traditional ways of estimating the cost such as estimating resource and schedule.

DISCUSSION