Software Assurance Objectives Hierarchy
Top Objective: Software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle, and that the software functions in the intended manner.

Strategy: Assure that software is of high quality and that it operates safely and reliably.

Objective: Software conforms to functional intent and can be assured to performs as planned (1)

Objective: Software system is robust and tolerant to failure, off nominal conditions (2)

Objective: Software does not adversely impact safety and contributes to system safety (3)

Objective: Software is secure and not vulnerable to compromise from internal or external influences [placeholder] (4)

Context: Applies generically to V&V and IV&V Efforts.
Objective: Software conforms to functional intent and can be assured to perform as planned (1)

Strategy: Achieve a high level of process maturity to ensure a robust software product (1.A)

Objective: Software development process minimizes insertion of errors (1.A.1)

Strategy: Assure a maintained, comprehensive working software development process exists (1.A.1.A)

Strategy: Assure software development process is followed and not introducing errors (1.A.1.B)

Objective: Faults, defects, or other issues have been found and resolved as part of the development process (1.B.1)

Strategy: Verify, validate and inspect throughout development to assure problems are found (1.B.1.A)

Strategy: Identify causes of anomalies (1.B.1.B)

Strategy: Resolve problems and verify solutions (1.B.1.C)

Objective: Remaining or known issues have been closed out to an acceptable level of risk (1.B.2)

Strategy: Track, address, and trend issues via a closed loop problem resolution process (1.B.2.A)

Objective: Faults, defects, or other issues have been found and resolved as part of the development process (1.B.1)

Strategy: Identify and resolve faults throughout the development process (1.B)

Strategy: Identify causes of anomalies (1.B.1.B)

Objective: Software V&V assures confidence in the interim and end software products (1.C.1)

Strategy: Assure a maintained, comprehensive working software V&V process exists (1.C.1.A)

Strategy: Assure software V&V procedures are complete and Nominal and off nominal conditions are addressed (1.C.1.B)

Strategy: Assure execution of V&V planning and procedures (1.C.1.C)

Strategy: Assure software product deliveries and final as-built configuration (1.C.1.D)

Context: Applies to full lifecycle from systems requirements to retirement

Context: V&V includes reviews, inspections, testing, demos
Objective: Software system is robust and tolerant to failure, off nominal conditions (2)

Strategy: Assure that software is developed in a robust manner, which decreases/eliminates errors, and includes necessary design features to prevent failures and off-nominal conditions from compromising ability to accomplish mission objectives (2.A)

Objective: Areas of weakness are known and addressed to the proper level (2.A.1)

Strategy: Participate in system analysis and design to determine areas of weakness, suggesting where software design can improve system operability (2.A.1.A)

Strategy: Identify, classify, collect, trend and report defect metrics (2.A.1.B)

Strategy: Analyze metrics, requirements, and design to predict and address problem areas (2.A.1.C)

Strategy: Analyze for software vulnerabilities and recommend design strategies (2.A.1.D)

Objective: Has multiple means of accomplishing functions that are critical to mission objectives (2.A.2)

Strategy: Determine appropriate level for functional redundancy between system and software (2.A.2.A)

Strategy: Assure appropriate software functional redundancy (2.A.2.B)

Strategy: Design in appropriate software architecture to meet and maintain critical functions (2.A.2.C)

Objective: Pathways for fault propagation or combination through software interfaces are limited (2.A.3)

Strategy: Design in protective barriers to prevent software from propagating system faults (system includes software, hardware, and human interface) (2.A.3.A)

Strategy: Design in protective barriers to prevent software from propagating system faults (system includes software, hardware, and human interface) (2.A.3.A)

Objective: Software is able to recover from anomalies affecting functionality (2.A.4)

Strategy: Provide fault management (detection, isolation, recovery) capabilities (2.A.4.A)