CAPSTONE: A Methodology to Assess the Safety Risks of Software-controlled Systems

William M. Teppig Jr.
Elyse Loosararian

BAE SYSTEMS
INSPIRED WORK

Precept One

- The core basic unit of an active system can be defined as:
  energy release to achieve a purpose,
  intended or not intended.

- Release can be enabled by:
  - mechanical action,
  - human action,
  - electrical action (including software), and/or
  - environmental influence.

Precept Two

- At its basic level, Command Software (CS) consists of:
  nothing more than **pre-stored intent**.
- *Therefore,* Command Software algorithms must be provided timely accurate assessments:
  - of sensor interpretations
  - of "State Conditions"
  - that must transfer through a series of interface "boundaries."

Precept Three

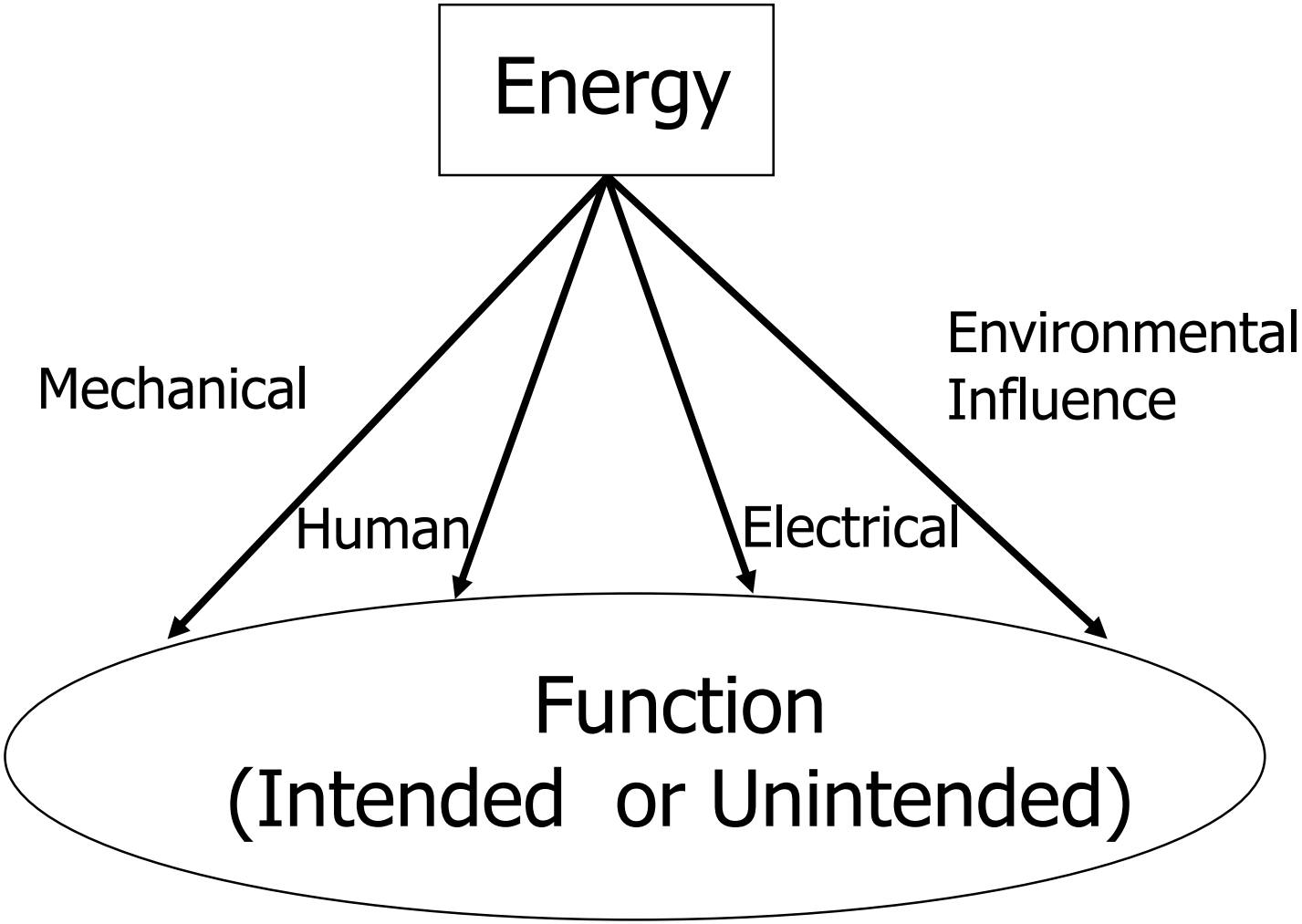
- A **system's architecture** establishes the maximum level of safety possible because state condition awareness is limited by:
  - the sensors selected,
  - the state conditions monitored,
  - the monitor rate, and
  - the methodology integrated to inhibit Top-Level Mishaps (TLMs).

- Just like "quality," safety cannot be "tested" into a system.

Precept Four

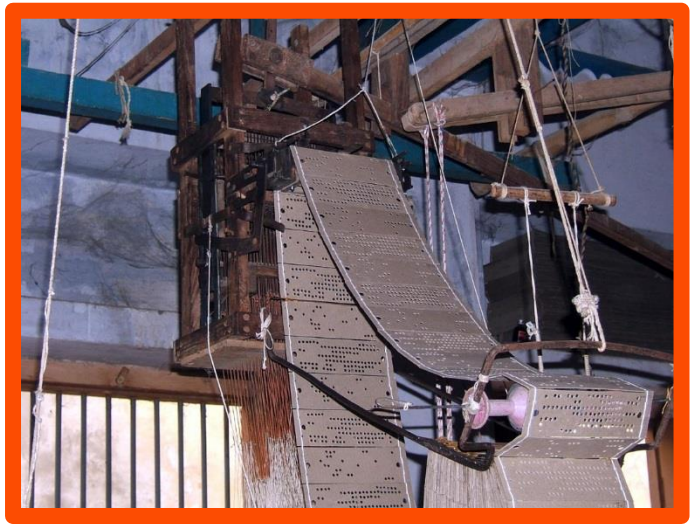
- A failure is NOT REQUIRED to place a system in an "Unknown, Unsafe State"!
- Design shortcomings can cause a system to be in "Unknown, Unsafe State" at start up!

Basic System Element



Software "Folklore Statement"

Stating that *software never fails* is the same as saying the "**holes go all the way through**" in these first *command control systems*!



BAE SYSTEMS
INSPIRED WORK

Software "Folklore Statement" Critique

- A more accurate statement could be:
  - The **designers failed** to anticipate applicable state conditions, so as to pre-store the desired reaction
  - The **designers failed** to provide applicable sensors to allow rigorous state condition monitoring of critical conditions
  - The **designers failed** to interrogate the sensor suite in a timely manner to preclude the formation of a TLM "tipping point"—*i.e.,* the BIT/BITE rate is disassociated with the shortest life cycle progression of the TLMs
  - The **designers failed** to provide robust, timely inhibit methodology to preclude the formation of a TLM "tipping point"
  - The **designers failed** to establish safe "norms" of action for the chosen effectors that would allow "anticipation" of the formation of a TLM "tipping point"
  - The **designers cannot** directly link their design risks with the selected remedial actions to control the system TLMs

System Safety Background

- Historical attempts at system safety are insufficient for modern systems
  - Traditional safety analyses were developed to analyze hardware dominant systems where the smallest unit (tolerance) had a proportional impact on the system output and paralleled the development of the system
  - In software dominant systems, the smallest unit (Bit) may have total control of the system output and few modern systems are developed from a “blank sheet”

System Safety Background, cont'd

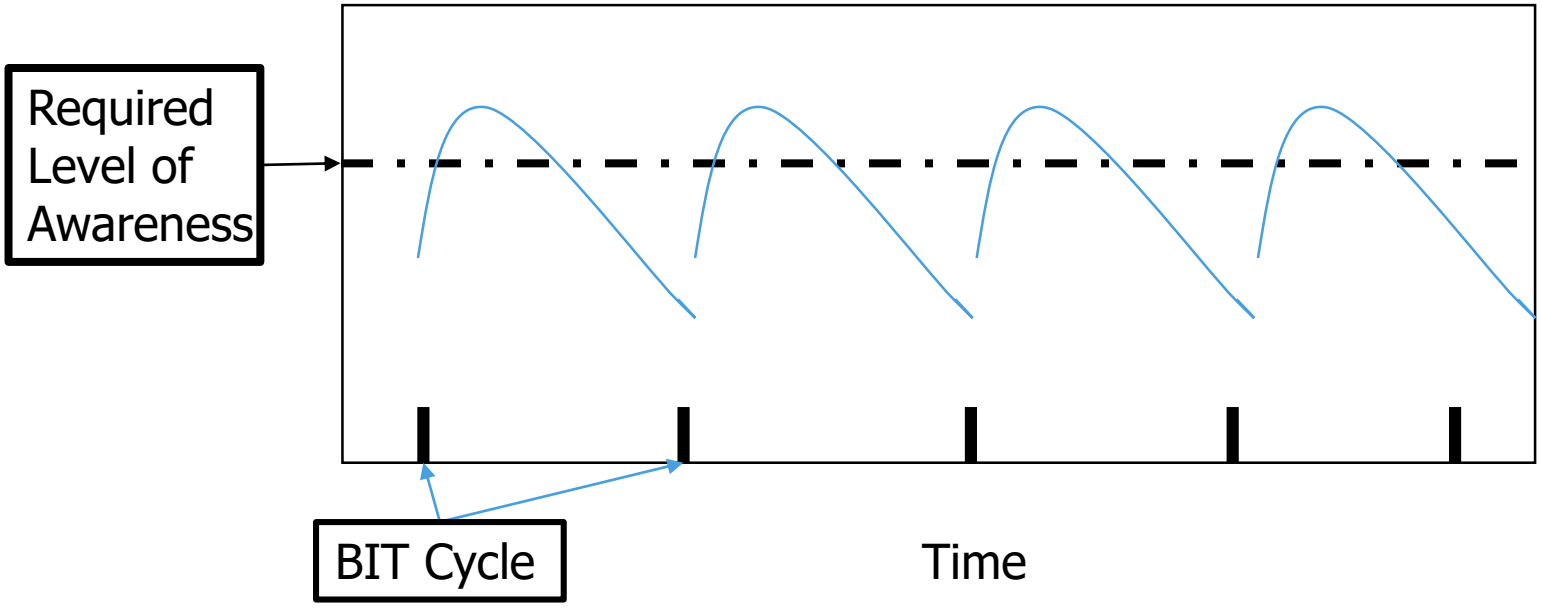
- Previous system safety methods are insufficient for modern software systems
  - Traditional safety analyses were developed based upon static "models" of the system design and notional performance of components and subsystems
  - Traditional safety analyses were based upon assumptions of isolation/independency between the system components to simplify the calculations required
  - The current construct is unable to take advantage of the current processors that can analyze multiple, concurrent, interactive variances of the constituent components-many of which have an "unknown" history or design (COTS, NDI, Legacy)
  - State Condition monitoring *RATE* was established before Top-Level Mishap Life Cycles were assessed for the final design

Approach for Autonomous Systems

- The proposed approach analyzes the robustness of the system's **architecture** at each of the defined system boundaries to:
  - assess the ability of the system design to monitor state condition changes, errors and failures *in a timely manner*
  - increase the probability of early discovery/detection of the formation of an impending hazardous state condition
  - begin early initiation of remedial actions
  - preclude the fault condition from cascading between the boundaries resulting in an inevitable mishap (tipping point)

State Condition Knowledge

- State condition awareness is at its maximum when Built-In Test (BIT) indications are provided to the Command Software and state condition status is established
- The certainty of that awareness degrades between BIT cycles relative to the shortest TLM Life Cycle



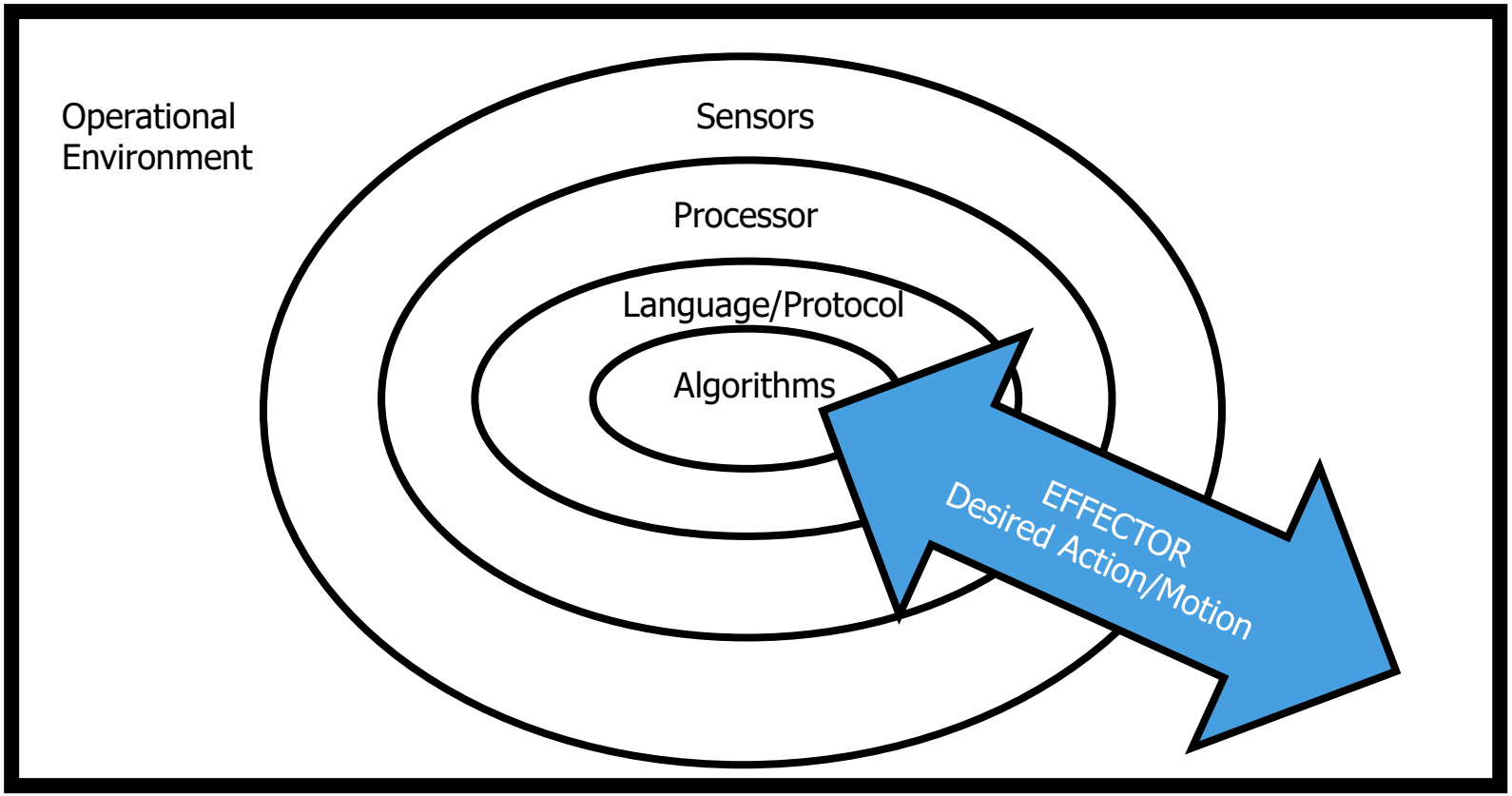
State Condition Definitions

- Safe State
  A condition in which knowledge and control of the system is maintained, at a sufficient level, to preclude the formation of a tipping point for any mishap
- Unsafe State
  A condition in which state condition control of the system is not maintained to inhibit reaching the tipping point for any mishap
- Known State
  When the system state is confirmed—sufficiently
- Unknown State
  When the system state is not or cannot be confirmed—sufficiently
- Unknown, Unsafe State
  When the system state is not or cannot be confirmed sufficiently in time to prevent reaching the tipping point for a mishap
- Unknown, Safe State
  When the system state is not or cannot be confirmed sufficiently, but the system state conditions are maintained to inhibit reaching the tipping point for any mishap

Actionable **Definition of Rigor** for Command Software Systems

- Each boundary between domains can allow errors of omission and commission
- State condition information is interpreted and displayed to the operator, whose actions are interpreted and provided to the system as another state condition parameter—with the same risk of errors
- Definitive control over the *state condition interpretations* is the basis of establishing the level of "rigor" for a system design
- Intrusive state condition queries controlling and monitoring the "pre-stored" intent performance provides a measurement of the degree of rigor within the system

Command Software System Domains



Autonomous System

BAE SYSTEMS
INSPIRED WORK

MIL-STD-882E **Lack of RIGOR**

- MIL-STD-882E requires a **subjective linkage** be established:
  - The ***degree* of functionality control** over the "actions" of the effector to safely project the intended response pre-stored by the designers!
  - The ***degree* of autonomy** given the command algorithms to control over the "actions" of the effector to safely project the intended responses pre-stored by the designers

MIL-STD-882E **Lack of RIGOR**, cont'd

- This "autonomy" is based upon a series of unsupported assumptions:
  - *"This definition includes the control of moderately complex system/software functionality, no parallel processing, or few interfaces, but other safety systems/mechanisms can partially mitigate.* ***System and software fault detection and annunciation notifies the control entity of the need for required safety actions****."*
  - *"This definition assumes that the safety-critical display information may be time-critical,* ***but the time available does not exceed the time required for adequate control entity response and hazard control****."*
  - *"This definition* ***assumes that there is adequate fault detection, annunciation, tolerance, and system recovery to prevent the hazard occurrence if software fails, malfunctions, or degrades****…"*

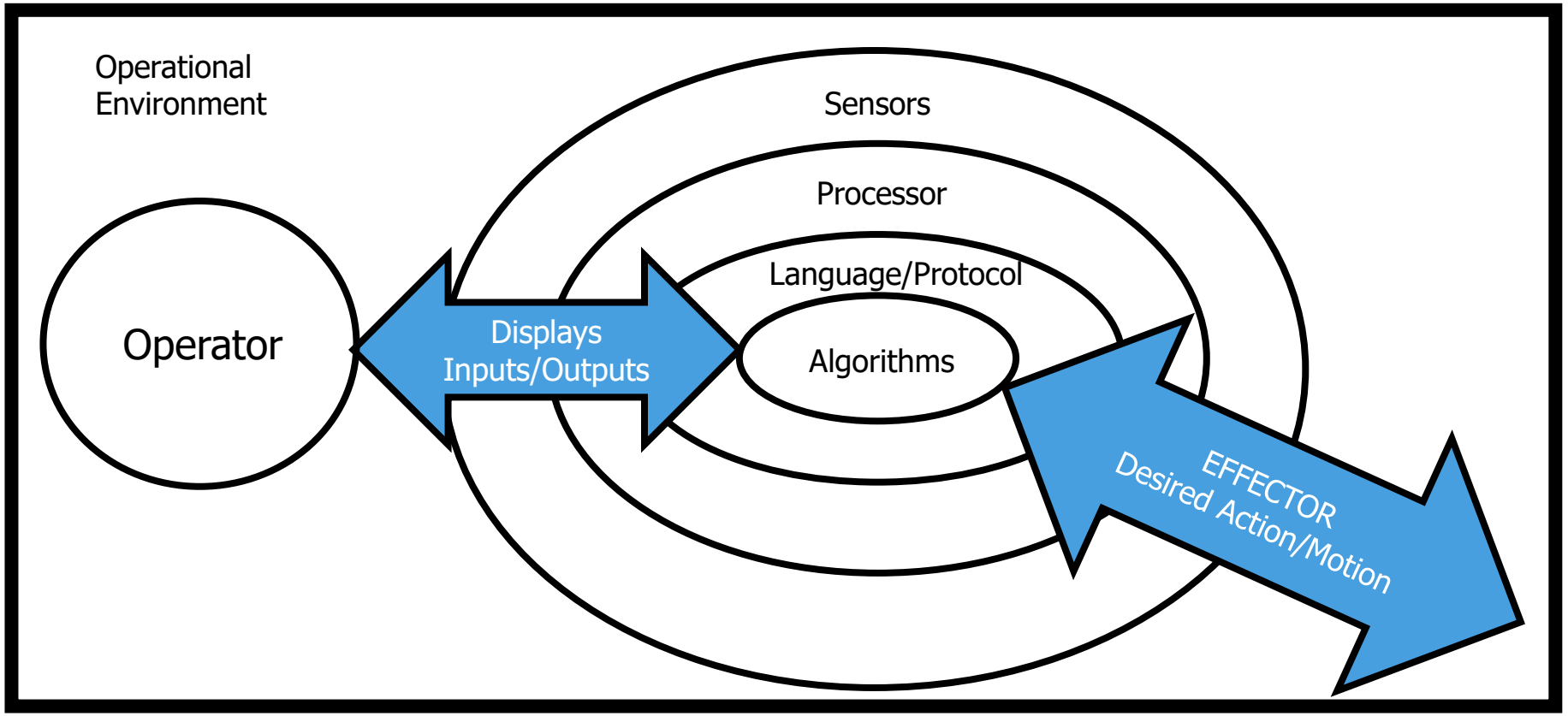
MIL-STD-882E **Lack of RIGOR**, cont'd

- The lack of RIGOR in establishing design precepts for adequate monitoring of timely, state condition dynamics for the pre-stored effector performance is amplified by:
  - Prescribing a series of code analysis assessments based on a poorly defined term "Level of Rigor Tasks"
  - In fact, "Rigor" is undefined in actionable terms that can provide Program Managers a level of confidence that the true risks are being controlled proportionally with the assets dedicated to resolving "software" hazards
  - Failing to provide guidance to re-shape the state conditions awareness performance to improve the level of safety

Approach for Semi-Autonomous Systems

- A "semi-autonomous" system (Man-In-The-Loop) can be assessed by this same methodology
- The **operator** adds another layer of two-way interface boundaries and required interpretations of "real-world" status conditions by the operator.
- The operators reactions will be driven by the "pre-stored intent":
  - *interpretations* for the system's state condition
  - *presentations* of the system's state condition
  - *interpretations* of the operator's inputs/intentions
  - *projection* of those inputs/intentions through a designated effector
  - *ability to monitor, interpret, measure, summarize and display* the final state condition of those operator inputs/intentions

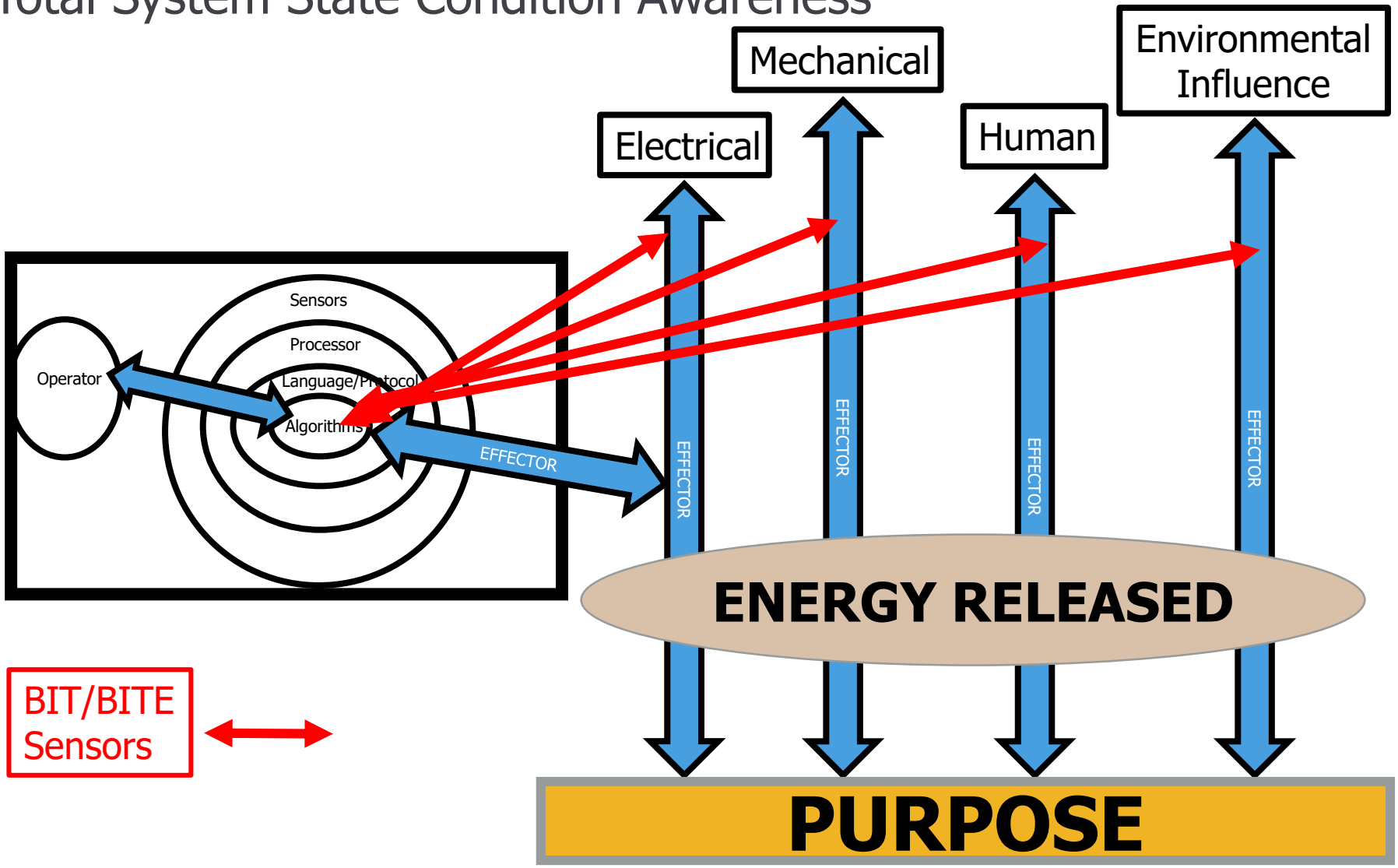
Command Software System Domains



Manned System

BAE SYSTEMS
INSPIRED WORK

Total System State Condition Awareness

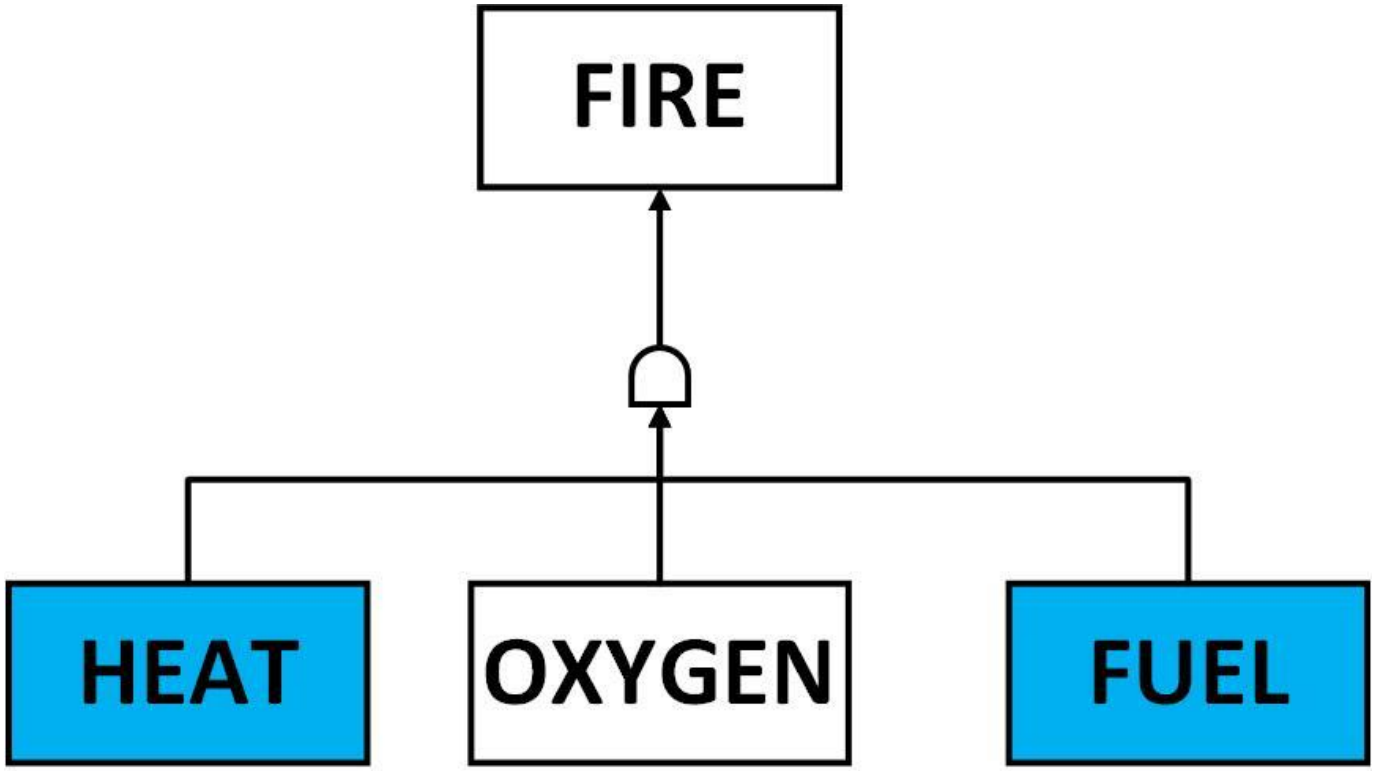


BAE SYSTEMS
INSPIRED WORK

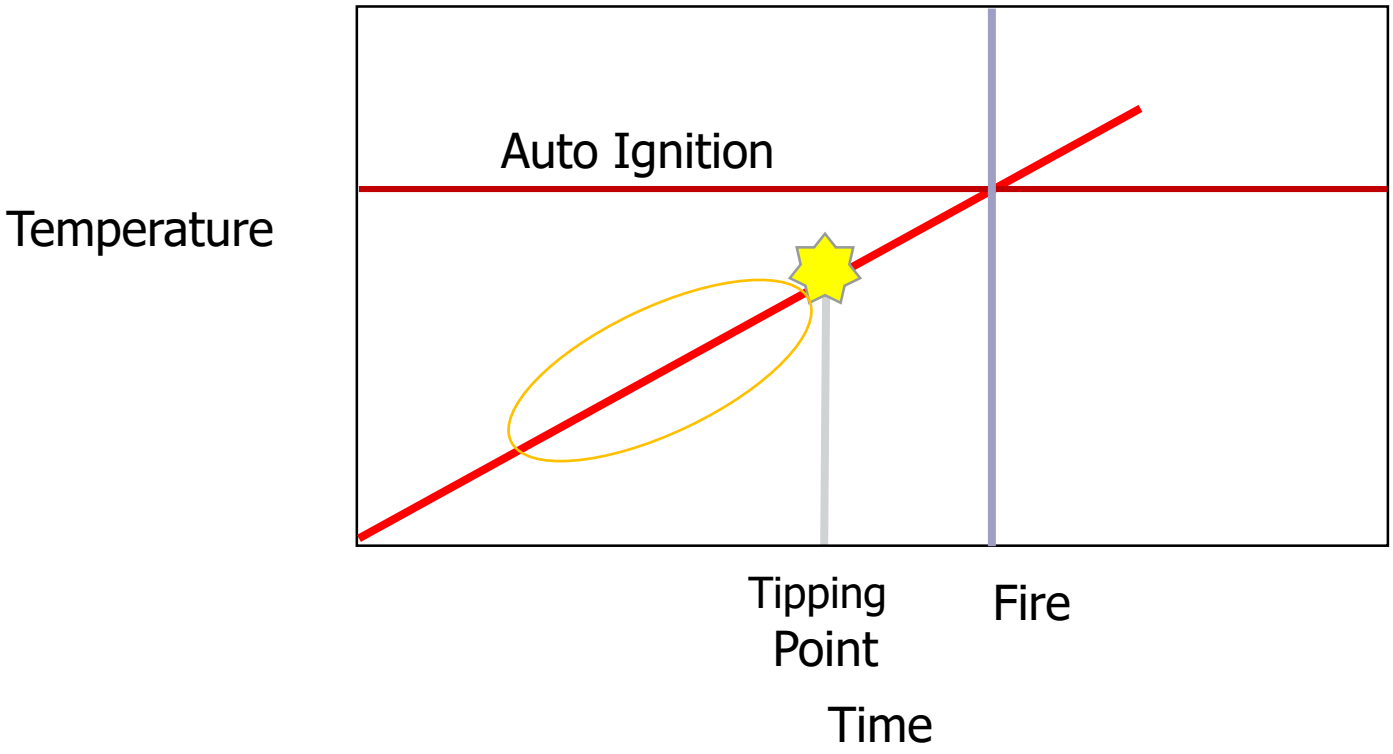
Total System State Condition Awareness

- Sensors, attuned to the Top-Level Mishap precursor conditions should be installed within the system to detect the energy release mechanisms that control effector functionalities
- These sensors should provide the software control algorithms with sufficient state condition information to execute the pre-stored intent to maintain the system in a Known Safe State
- Any extrapolations and projections required to implement remedial actions "in time" to preclude the formation of a TLM tipping point reduce the fidelity of the control algorithms awareness of actual system state conditions and reduces the confidence (rigor) of the system's level of safety

Top-Level Mishap Precursor Conditions



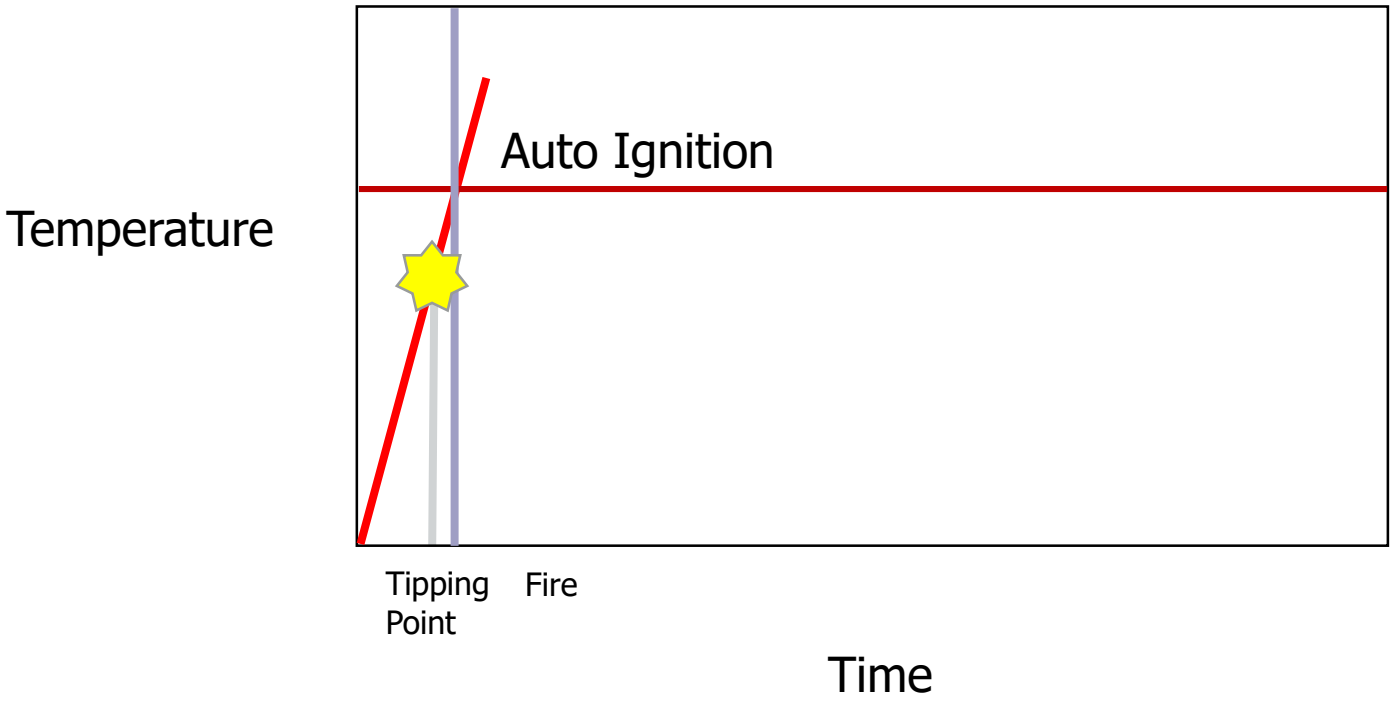
Top-Level Mishap Precursor Conditions



- Oxygen level must be adjusted before the tipping point is reached
- Time for projected state condition—initiate oxygen adjustment

BAE SYSTEMS
INSPIRED WORK

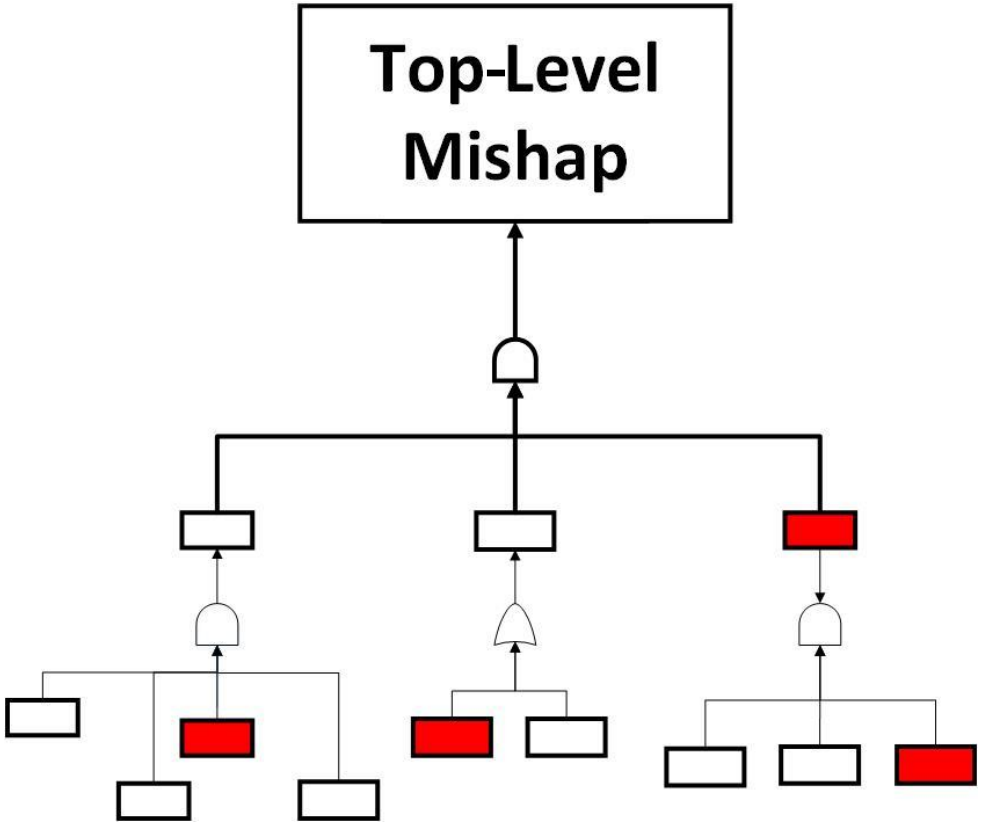
Top-Level Mishap Precursor Conditions



- Oxygen level cannot be adjusted before the tipping point is reached
- NO Time for projected state condition to initiate oxygen adjustment
- MUST use a fast acting fire extinguishant

BAE SYSTEMS
INSPIRED WORK

Top-Level Mishap Precursor Conditions



Emerging pattern of unique state conditions unfolds to provide sufficient confidence that an "early start" on remedial actions is required by projecting an impending tipping point

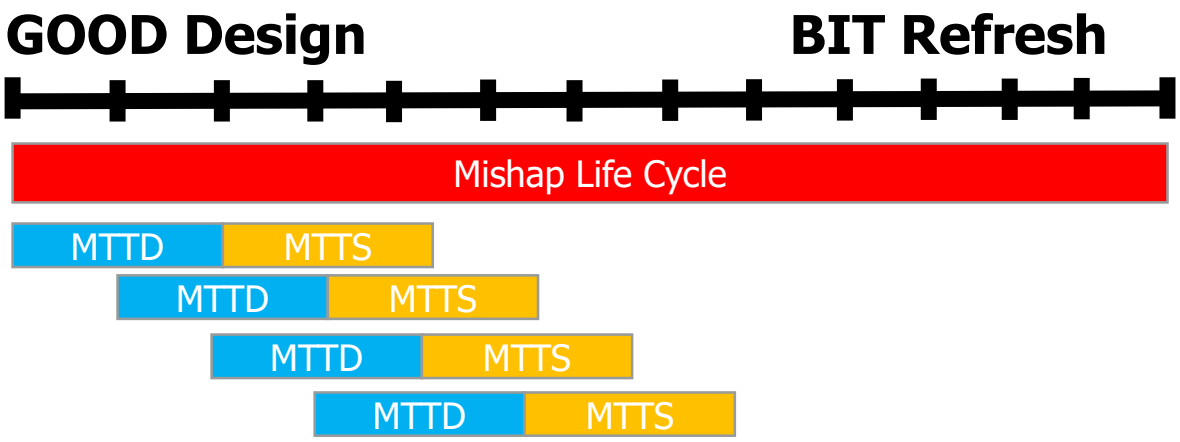
Critical Timeline

- Mean Time values were selected as a nominal measure for this presentation. The more critical the Top-Level Mishap impacts, the less time should be allowed for this events.
- Mean Time to Detect (MTTD)
  - Average Time to recognize emergence of all required TLM initiating event(s)
  - TTD_{CRIT} – CRITICAL Time to Detect is the last query action (BIT cycle) before time required to complete Time To Stop *PRIOR* to achieving the Tipping Point
- Mean Time to Stop (MTTS)

  Average Time to respond confirmation of a TLM event and the declaration of requirement to initiate STOP actions
- Top-Level Mishap (TLM) Life Cycle

  Time from the emergence of ALL required causal factors of the TLM and the Tipping Point

Basic Critical Timeline Examples

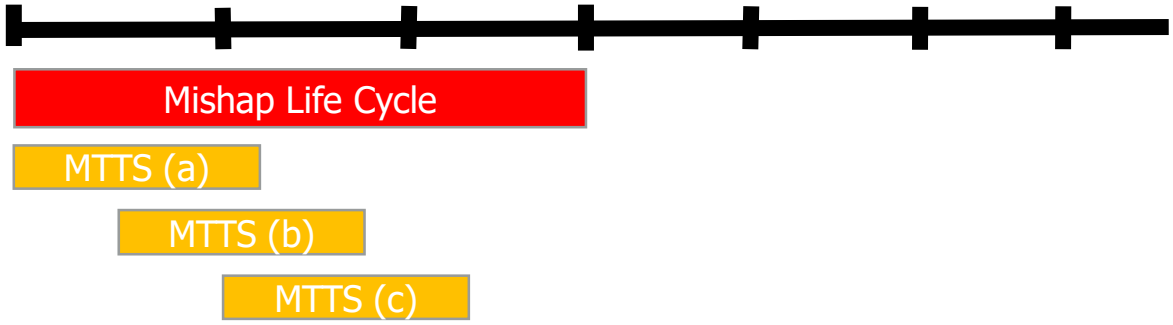


The sum of MTTD and MTTS is well within the Mishap Life Cycle allowing multiple opportunities to place the system in a known, safe state before damage/injury

Basic Critical Timeline Examples, cont'd

BAD Design and BAD Architecture

BIT Refresh



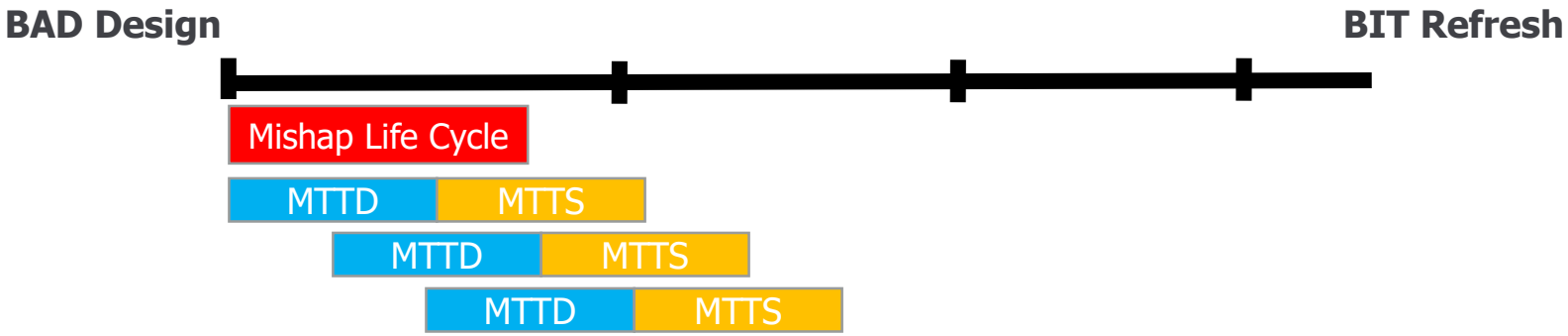
The system architecture does NOT allow the "Pre-stored Intent" to sense the onset of the mishap life cycle (**No detection capability**)

Making it IMPOSSIBLE to save the system from injury/damage- even with three control methods to stop within the life cycle of the mishap

The system is immediately in an UNKOWN-UNSAFE State– without a failure occurring!

Options: Redesign the sensors suite or the projection queries monitoring the causal elements

Basic Critical Timeline Examples, cont'd



The sum of MTTD and MTTS is longer than the Mishap Life Cycle
Making it IMPOSSIBLE to save the system from injury/damage

The system is immediately in an UNKOWN-UNSAFE State—without a failure occurring!

Options: Shorten the MTTD and/or the MTTS and/or the BIT refresh rate or the projection queries monitoring the causal elements

or

Reduce the probability of occurrence of the event through traditional system safety methods by developing/inserting a blocking element (AND GATE) to control the development of Top-Level Mishap

Probability of Detection

- Limited by Command System Design **Architecture**
  - BIT/BITE
    - Sequencing
    - Periodicity
    - Redundancy
    - Exhaustiveness/intrusiveness
- Sensor Characteristics
  - Location (time and location in design)
  - Accuracy
  - Reliability
  - Communication Surety
- Remedial Actions
  - Effectiveness
  - Timeliness
  - Overtness (for operator and system)

Probability of Detection, cont'd

- "Scenario Unique" based upon:
  - Top-Level Mishap Formation Precursors (Causal Elements)
  - Component Performance
  - Operator Performance
  - Environmental Characteristics
  - Variability within each of the above characteristics
  - Detection can be improved by establishing nominal performance profiles for projected system responses and environmental norms
- Broad categories of magnitude demarked by two Command System design architecture elements
  - State Sensor Performance
  - State Condition Query Performance/Projection of the TLM Causal Elements

Design Options to Increase Safety

- Given:
  - TLMs have been evaluated to determine:
    - Causal Elements supporting critical events and threshold profiles
    - Life cycle characteristics
      - Max/min life cycles
      - State condition change profiles for all contributory processes
      - Situational variables that influence the state condition rate changes
  - Sensors
    - Have an established level of accuracy and reliability
    - *Actually* are installed to accurately, directly sense the state condition change profiles for all contributory processes
  - Built-In Test cycle time is short enough to allow multiple updates to the "Known State" before a tipping point can be reached

Design Options to Increase Safety, cont'd

- Command Software
  - Monitors the sensors and can respond before a tipping point can be reached
  - Is programmed to provide anticipatory actions allowing a declaration of a TLM and to initiate "Time To Stop" efforts before the TLM is manifested
    - Through the use of notional performance profiles of the causal elements
    - Through the use of state condition change energy monitoring, (electrical power, hydraulic power, mechanical bending, heat, etc.)
    - Through state condition change rates to prioritize remedial actions to match the preferred and emergency remedial actions- WITH the available or project available time to successfully respond

Probability of Software Command Failure

- The system architecture can provide a number of measureable design safety features that are deterministic on the probability of a potential tipping point being formed un abated for a TLM.
- A measurement tool assessing the operational availability of the *sensing function* being available (assuming it is installed to actually, directly sense the precursor state condition events that support the formation of a tipping point) can be developed.
- The life cycle of the TLM incorporating all of the variables of the precursor state condition events that influence the Max/Min times to detect or times to stop.
- The number of BIT pulses within that established Max/Min times to detect or times to stop. (the more inspection cycles, the higher the probability of detection)

State Sensor Performance Parameters

I - Maximum

- Redundant sensors
  - Acting in opposite states (Not open and Closed, Open and Not Closed)
  - Distributed along the motion path
- Sensors utilizing different attributes
  - Visual tracking
  - Heat profile
  - Magnetic movement
  - Physical contact
  - Electrical resistance/continuity
- Unique internal identification of each sensor

State Sensor Performance Parameters, cont'd

II - Minimum

- Single sensor
  - At stowed potion or Full open
- Single Sensors utilizing one attribute
  - Visual tracking
  - Heat profile
  - Magnetic movement
  - Physical contact
  - Electrical resistance/continuity
- No sensor identification or inferred by I/O channel
- State condition inferred by "summed" electrical circuits with "continuity" interpreted as a state condition

State Sensor Performance Parameters, cont'd

III - Least

- Single sensor
  - Change of "position" interpreted as another state condition
  - No timing requirement
  - No sequence of events requirement
  - No "reasonableness" query

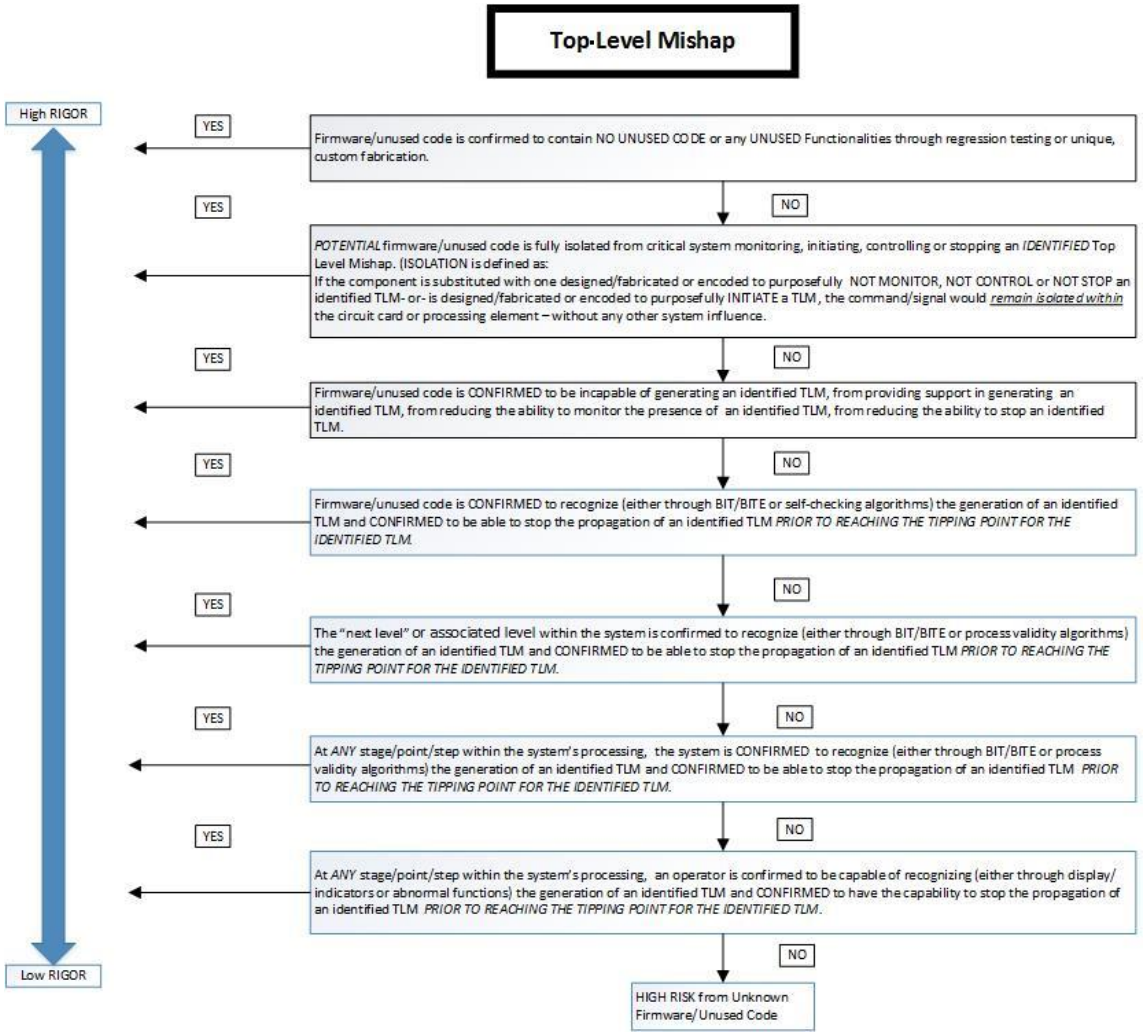
IV - None

State condition "assumed" to occur after command has been given

Sample Command Software Safety Parameters

Power On/Off	Inductive Power Monitoring - (Presence, Polarity, Magnitude)
Loss of Communication - monitoring at sufficient rate to detect *IN TIME*	Redundant Sensor Scheme - with incremental placement
Unique Machine ID for Components	External Actuator State Monitor(s)
Pre-operational BIT - sequenced FROM a known component to each succeeding level, before operate power becomes available for each functionality	Internal Actuator State Monitor(s)
Background BIT - able to function WHILE system is functioning	Change of Physics Used for Monitor(s) - (optical, heat, magnetic, electrical, weight, mechanical etc.)
Intrusive BIT - requires the system to be off-line	Opposite Acting Sensor Redundancy - • Stowed and Not Deployed • Deployed and Not Stowed
Notional Operational Power Profiles - *to PREDICT* potential deviations	Change of State History With Notional Time Profile
Notional Time To Complete Profiles - *to PREDICT* potential deviations	Sanity Checks for Reasonableness of outcomes and state conditions
Dual Actuators with Casualty Mode Defaults	

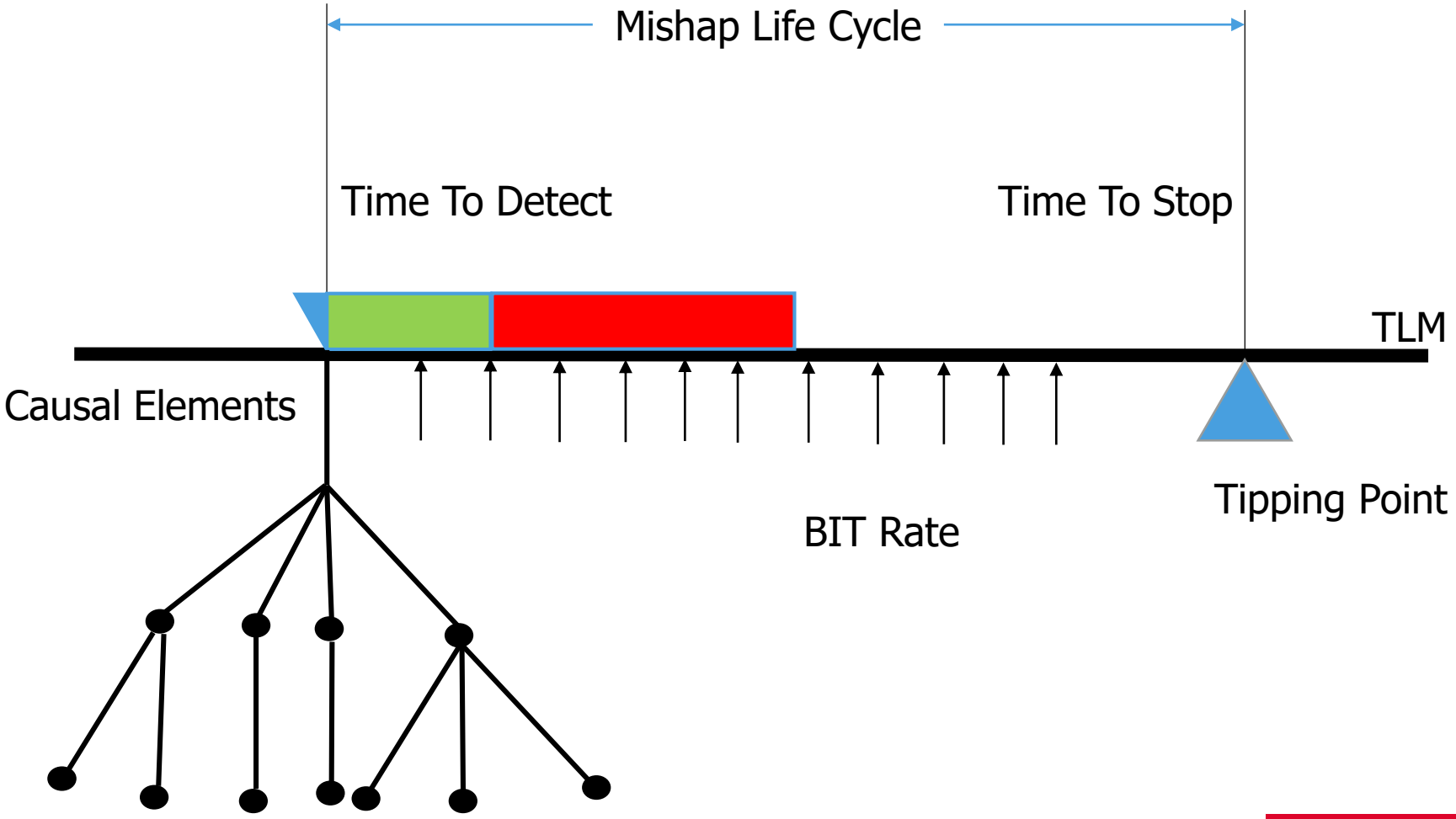
Sample Guidelines for Ranking State Condition Control **RIGOR** for Firmware/Re-use Code



BAE SYSTEMS
INSPIRED WORK

CAPSTONE

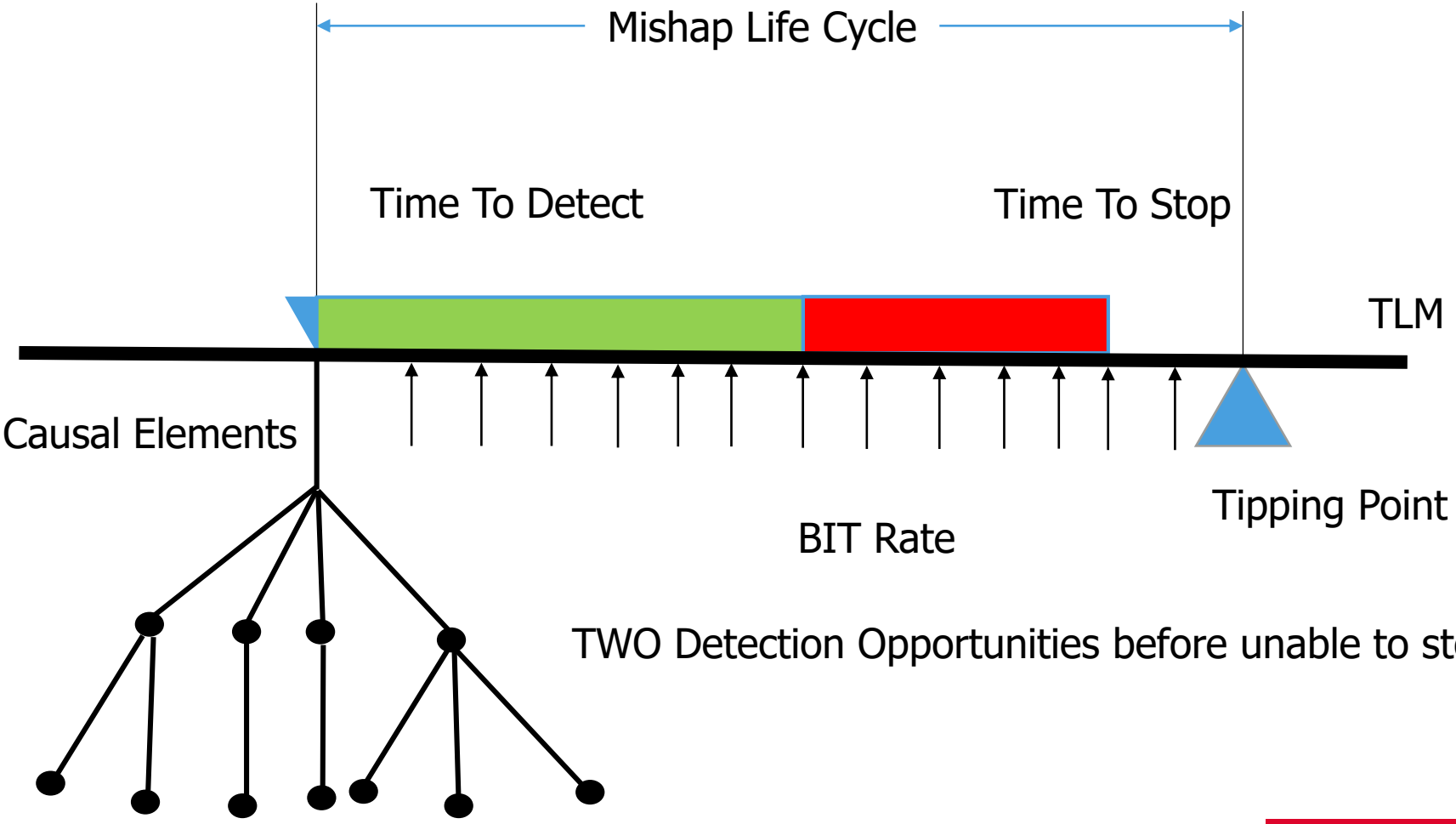
Optimum



BAE SYSTEMS
INSPIRED WORK

CAPSTONE

Acceptable

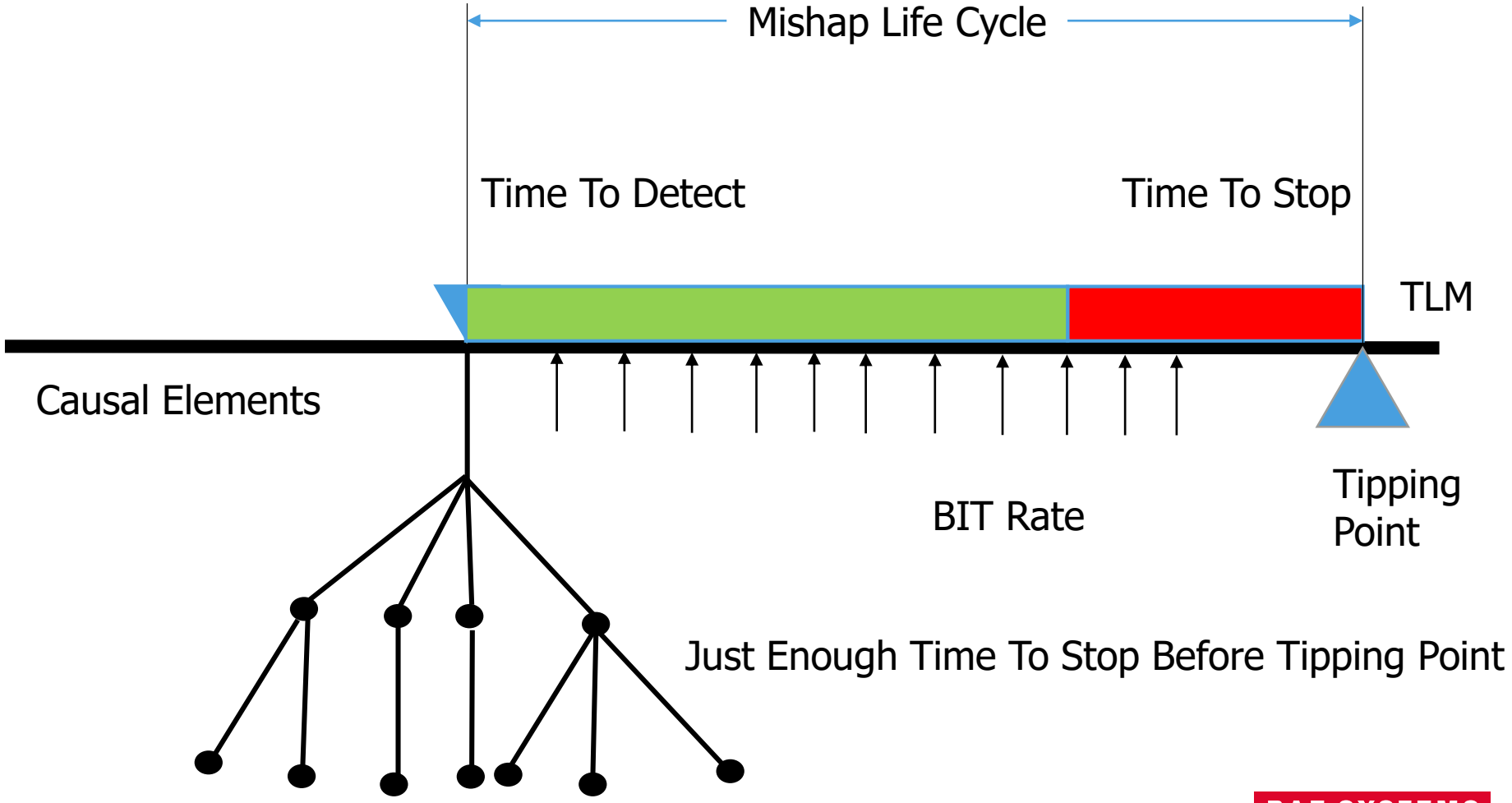


TWO Detection Opportunities before unable to stop

BAE SYSTEMS
INSPIRED WORK

CAPSTONE

Minimum

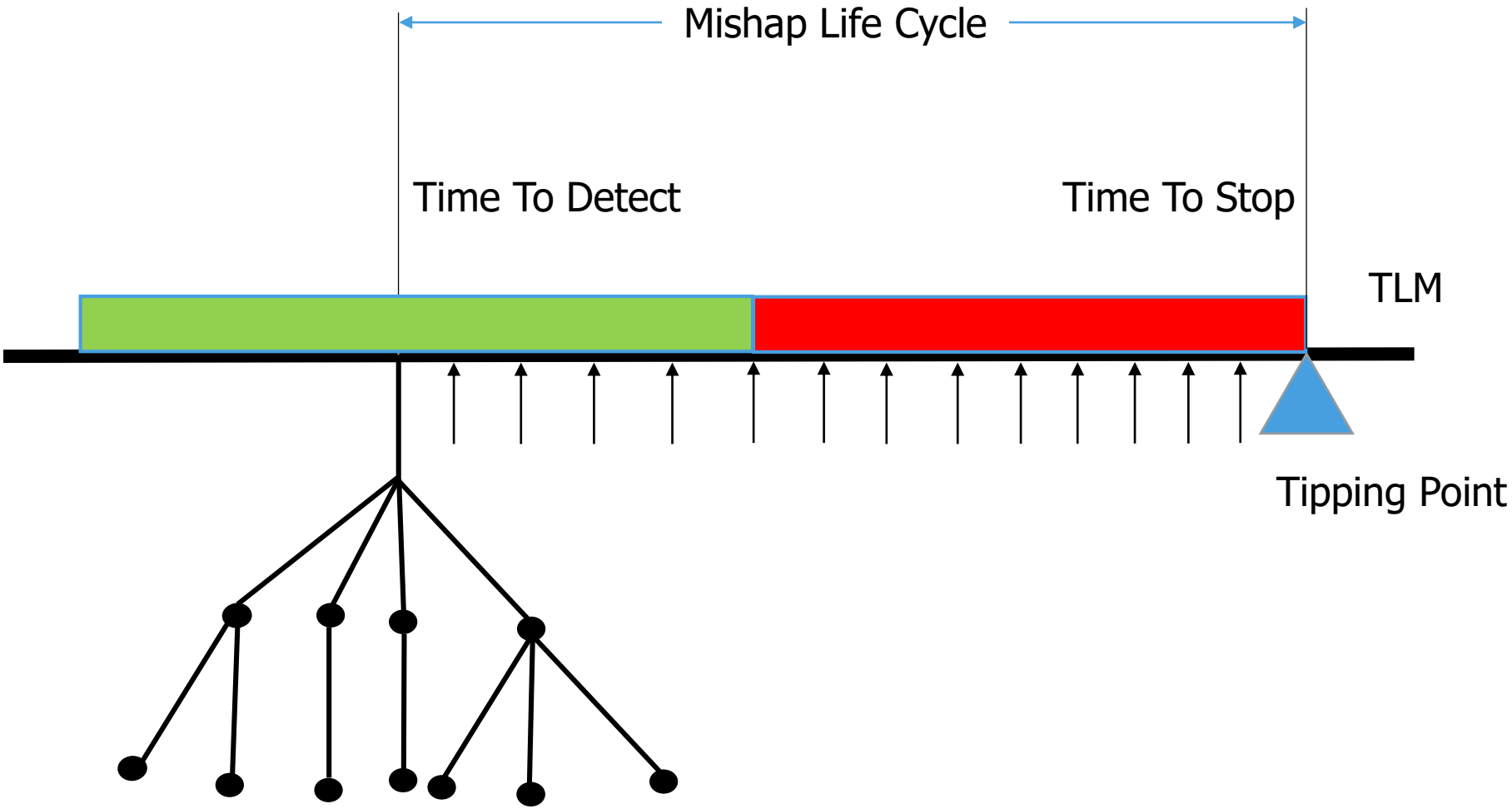


BAE SYSTEMS
INSPIRED WORK

When to Stop Stopping?

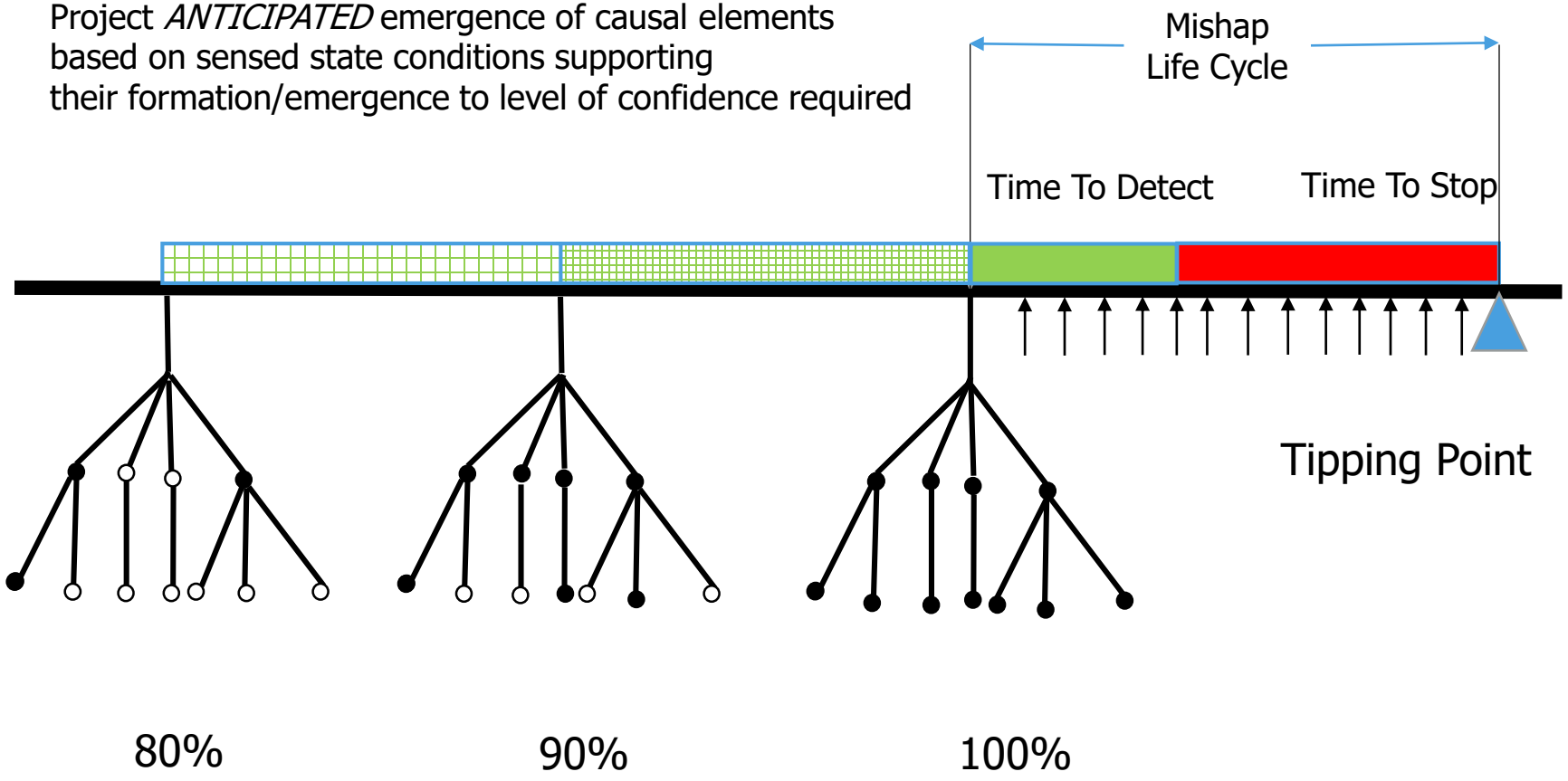
- State condition monitoring of the efforts to successfully STOP an impending TLM must be included in the design architecture
- Failure to STOP the projected Tipping Point may require the early initiation of Damage Control effects to reduce the overall impacts of the TLM
- Utilizing the same or additional sensors used to monitor the emergence of the causal elements will track the reduction of the causal element underpinning of the TLM
- Additional sensors may be required to monitor the effectiveness of complementary processes included to attenuate the progress of the TLM towards its Tipping Point
- Command software must monitor the effectiveness of both the reduction of the causal elements influence and the attenuation efforts to definitively assess the STOP efforts

What to do?



What to do?

Project *ANTICIPATED* emergence of causal elements based on sensed state conditions supporting their formation/emergence to level of confidence required

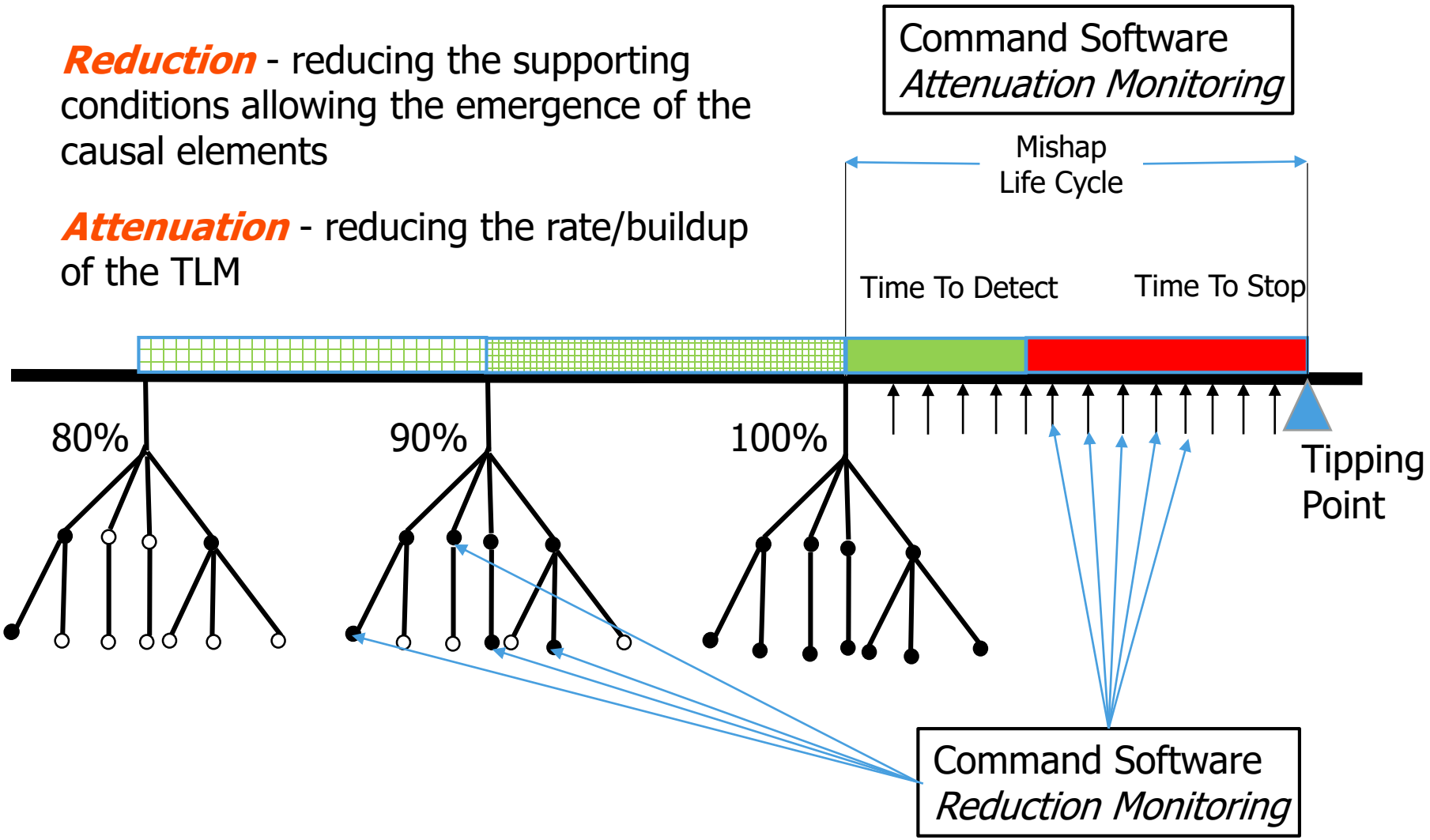


BAE SYSTEMS
INSPIRED WORK

When to stop stopping?

Reduction - reducing the supporting conditions allowing the emergence of the causal elements

Attenuation - reducing the rate/buildup of the TLM



BAE SYSTEMS
INSPIRED WORK

When to Stop Stopping?

Fire Example:

- Reduction in Underpinning
  - Reduction in temperature – fast enough
  - Reduction in oxygen percentage – fast enough
  - Reduction in fuel availability – fast enough
- Attenuation Effort
  - Water mist for cooling
  - HALON / FM 200 / PPK /AFFF
  - "Turn into the wind"
- Damage Control Efforts
  - Jettison over board
  - Deluge flooding
  - Abandon ship!

BAE SYSTEMS
INSPIRED WORK

Goal

- Design should have a continuously iterative nominal performance profile with constrained variances.
- If the Mean Time To Detect is excessive, then the detection interrogations (BIT/BITE) must be accelerated and the cycle times shortened
- If Mean Time To Stop is excessive, then nominal performance profiles must be developed to allow advanced notice of impending Tipping Points

The system architecture must be designed/redesigned to detect and stop the formation of Tipping Points before it is too late to stop!

CAPSTONE Utility During Acquisition Cycle

- ***CAPSTONE*** can:
  - assist in system architecture during CONCEPT Phase
  - establish the BIT/BITE requirements
  - determine the inherent level of safety
  - provide insight into potential failure modes/impacts
  - evaluate lower risks
  - provide remedial action guidance
  - determine adequacy of remedial efforts
  - provide acceptance OQE for risk acceptance throughout the life cycle of the system