_Validation and Verification for Mission and Life Critical Systems

Software development teams and project managers are constantly seeking accuracy and reliability – but when mission, life and safety critical systems (M&LCS) are introduced, the stakes become exponentially higher. M&LCS require defect-free software, and many teams in this field are looking to technology – like software cause-effect modeling (CEM) – to achieve it.

The Challenges Facing Mission & Life Critical Systems

Regulatory Environment

Development teams, regardless of which industry they're in, face an ever-growing number of standards and rules. With multiple governing bodies determining constraints and processes – and heavier regulations as criticality increases – staying on top of development while meeting ever-changing requirements is both labor-intensive and time-consuming.

Need for System Reliability

System reliability is among the most significant risk factors to safety critical systems. Although there are various ways to achieve this, the starting point is always a thorough understanding of intended system behavior, followed by comprehensive methods for testing.

Human Error

Perhaps the biggest liability in software development, human error poses a major concern to many mission-critical applications. Although humans are our greatest source of innovation and creativity, they're still susceptible to mistakes.

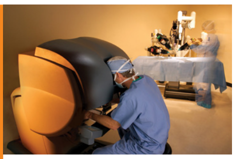
Project Overruns & Management Obstacles

Beyond the obvious implications of mission-critical applications, there are secondary issues that arise during an inefficient testing process. Costs and time to market skyrocket with multiple iterations and incomplete coverage, which have long-term implications on defect escapes and expense.

Mission & Life Critical Systems Are Everywhere

Although the specific challenges facing each group are unique, various industries share the common thread of software development needs with mission and life-critical systems:

Medicine



As technological advancements give medical professionals more time and freedom to take care of more patients, progressive systems – such as the da Vinci surgical robot – rely on complex software to control the system and protect human life.

Energy

The potential threat to both human life and the environment warrants a serious look into the energy realm. Ultra-high assurance systems such as nuclear reactors require the highest level of validation and verification available.

Automotive

Automotive systems – which are safety critical by nature – are becoming increasingly dependent on software. If that software fails, it can lead to injuries, loss of life and massive recalls.

Defense

Defense systems operate in life and death situations – and the safety of our armed service members is paramount. As computer systems become more central to these operations, their effectiveness is vital to our national security.

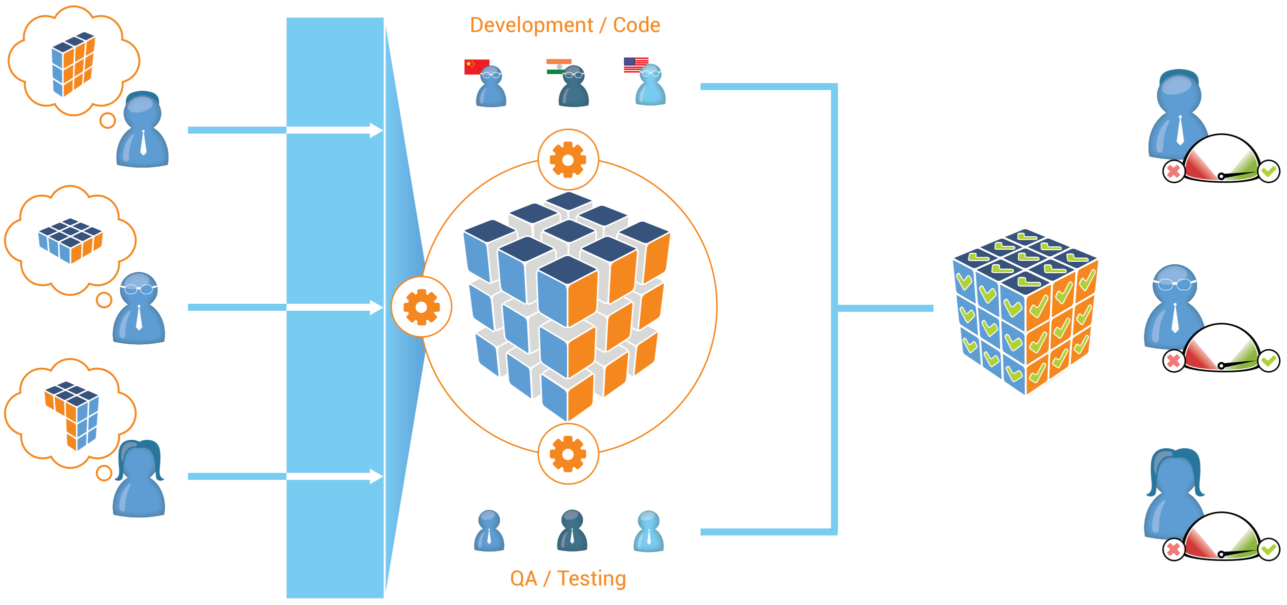
The United States' Unmanned Aerial Vehicle (UAV) programs are a tangible example of M&LCS development put to use. Pre-development cause-effect modeling could have prevented or mitigated most of the issues in the Hunter UAV program of 1988:

- *Distributed development teams*
- *Ever-changing requirements*
- *A weak test program based on fluid requirements*

Today, the need for defect-free output is arguably increasing as UAVs are considered for domestic use – subjecting them to even broader sets of regulations governing domestic aviation, such as DO-178C, which prescribes strict standards for functional test coverage.

The first step to a defect-free system is setting requirements and establishing a thorough understanding of intended behavior before development ever starts. By applying the formalism of Critical Logic's model-based testing tools to a set of comprehensive functional requirements, this "holy grail" of development can be reached.

Applying Critical Logic's CEM Platform Solution to Mission & Life Critical Software



Need for system reliability

Start with the end in mind – building a cause-effect model of intended system behavior not only resolves all requirements, but ensures from the start that 100% functional variation test coverage will be achieved.

Regulatory environment

Keep pace with changing regulations. When regulations evolve, so too can your software model – quickly, easily, and completely. Use your updated model to validate that your new requirements are compliant before you build the system; all while minimizing your time, effort and redevelopment cost.

Human error

Complement human intelligence and innovation with a system that removes sources of human error from your testing processes. Logical models demand complete requirements before testing, mathematical algorithms generate model-based test assets automatically, and test automation code is generated automatically from the model-based tests. At each step, modeling and automation reduce the opportunity for human error.

Project overruns/mismanagement

Help project managers perform their jobs more effectively through requirements that set measureable expectations and quantify intended results. When system behaviors are modeled first, validation is moved forward, development becomes more predictable, iterations are reduced, and adequate time remains for automated testing and full system verification.

**To learn more, contact Critical Logic at 415.814.9515
or visit us at www.Critical-Logic.com**