Systems Thinking in Fire Control Software Development
April 2014
Ross D. Arnold

UNCLASSIFIED
Distribution Statement A
- ARDEC Intro
- Intro to Fire Control Systems
  - Brief description
  - Examples
- Intro to Systems Thinking
- Systems Thinking in the SW Dev Process
  - Example Cases
  - Key Points
- Conclusion
- **U.S. Army Armament Research, Development and Engineering Center (ARDEC)**
  - Located at Picatinny Arsenal, NJ
  - Mission: Empower, unburden, and protect the Warfighter by providing superior armaments solutions that dominate the battlefield.
  - What does that mean?
    - Developing advanced weapons, ammunition, and fire control systems

- **Weapons and Software Engineering Center (WSEC)**
  - Sub-division of ARDEC
  - Software design and development
What is a Fire Control System?

- Software & hardware that enables:
  - Digital communications
  - Fire missions
  - Ballistic calculations
  - Point and shoot
  - Movement
- Purpose:
  - Digitize manual gunnery
- Applied to:
  - Towed Artillery
  - Mortars
  - Self-Propelled Artillery
Artillery Fire Control:

- M119 (Towed 105mm)
- M777 (Towed 155mm)
- Portable Excalibur Fire Control System (PEFCS)
- Paladin (Self-propelled 155mm)
Mortar Fire Control:

- Mortar Fire Control System (MFCS)
- Lightweight Handheld Mortar Ballistic Computer (LHMBC)
- Dismounted 120mm (MFCS-D)
- Precision Lightweight Universal Mortar Setter System (PLUMSS)
Systems Thinking Definition

- A set of synergistic thinking skills used to understand complex systems and predict their behavior.
  - See “wholes” and “parts” simultaneously
  - Understand how system structure causes behavior
  - Recognize interconnections and feedback loops
  - Predict dynamic behavior
  - Simplify through abstractive modeling
- Seeing the world “differently”
- Not intuitive
How does this relate to software development?

- Software dev process is a complex system
- Not to be confused with the software itself!
- Development process has:
  - Many elements
  - Interconnections
  - Feedback loops, delays
  - Uncertainty
Software Development System

- Is this accurate?
- All models are wrong, some are useful. (George Box)
Case 1: Code Reviews

- Traditional approach – mandate reviews
  - Policy resistance (classic systemic problem)
- What’s the system?
  - All developers “lazy?” No!
  - Humans like to perceive value
  - Recording results = context switching
- Solutions?
  - Improve information flow
  - Make recording easy
  - Many more, depends on system
Case 2: Information Flow

What’s wrong with this picture?
Case 3: Improving Productivity

“Crack the whip?” – No!

One Systemic Option: Minimize Distractions
- Context switching
- Fire-fighting
- Bugs
- Constant “urgent” requests

Best Practices
- Clear task
- Devote X time per day to task
- Target deadline
- Empowerment – devs know their purpose
Systems Thinking Key Points

- The software development process is a system
  - It has humans, humans are complex
  - Information flow
  - Lags and delays between decisions
  - Feedback loops, many of which might not be intuitive
- Problems require systemic investigation
  - There is no blame! (Senge 1990)
Systems are everywhere!

- Software development is no exception.
- Systems thinking is a transferable approach.
Questions?

Contact:
Ross Arnold
Towed Artillery Technical Lead
U.S. Army ARDEC
(973) 724-8618
ross.arnold@us.army.mil