IV&V Facility

Analyses of the Effects of Auto Generated Code on Independent Verification & Validation (IV&V) of Mission Critical Software

Noble N. Nkwocha
Andrew H. Sung, PhD
Qingzhong Liu, PhD

IV&V Facility

Important Facts from Prior Auto-Code Analyses on IV&V Projects

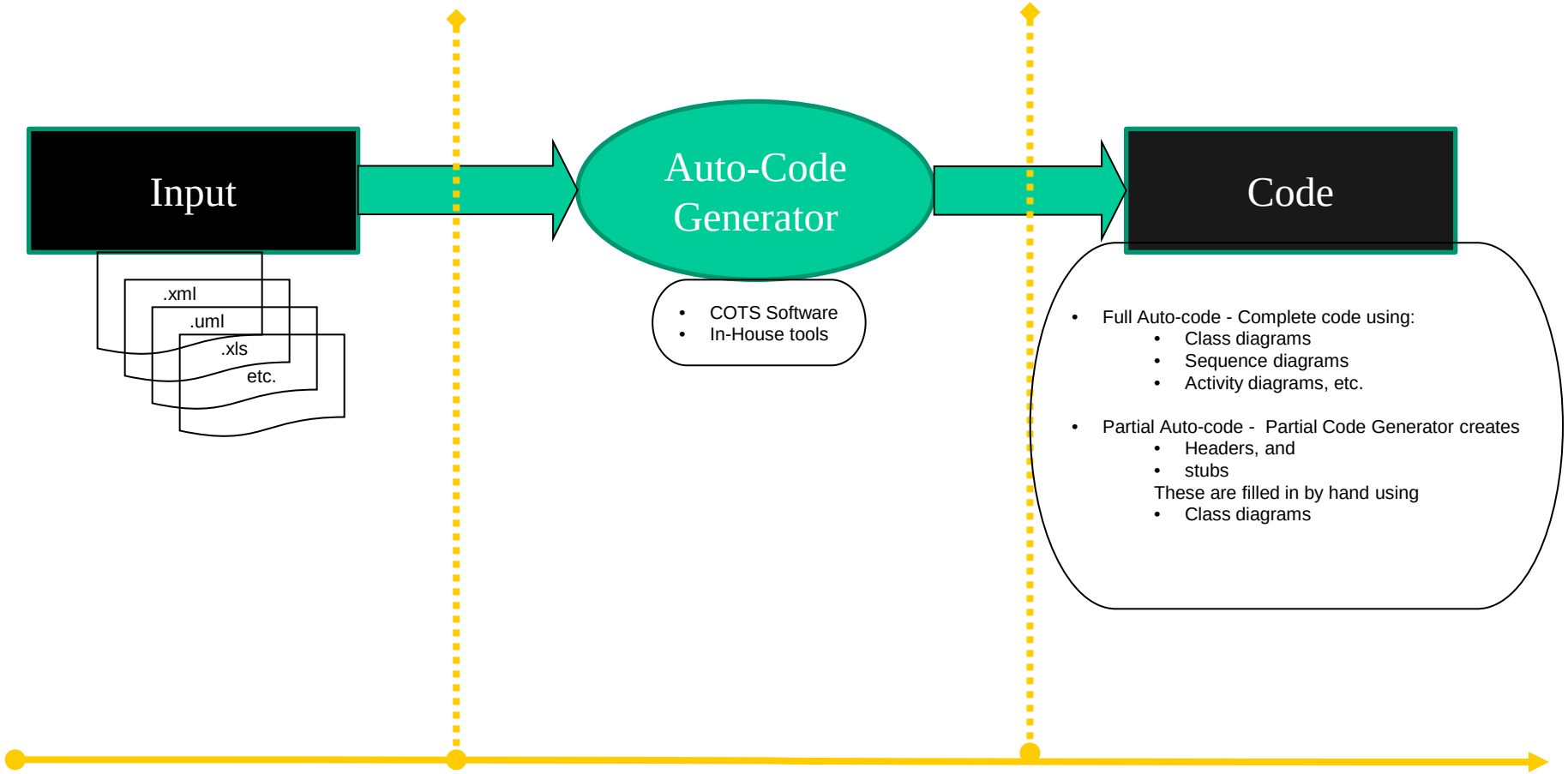
- Several, NOT fewer NASA IV&V Projects had / have Auto-Generated Code
- Question on Areas of IV&V Focus
- Higher SLOC Count of Auto-Generated Code – Compared to the equivalently hand-written code

*
IV&V Completed

IV&V Facility

How current IV&V methods treat Auto-Code

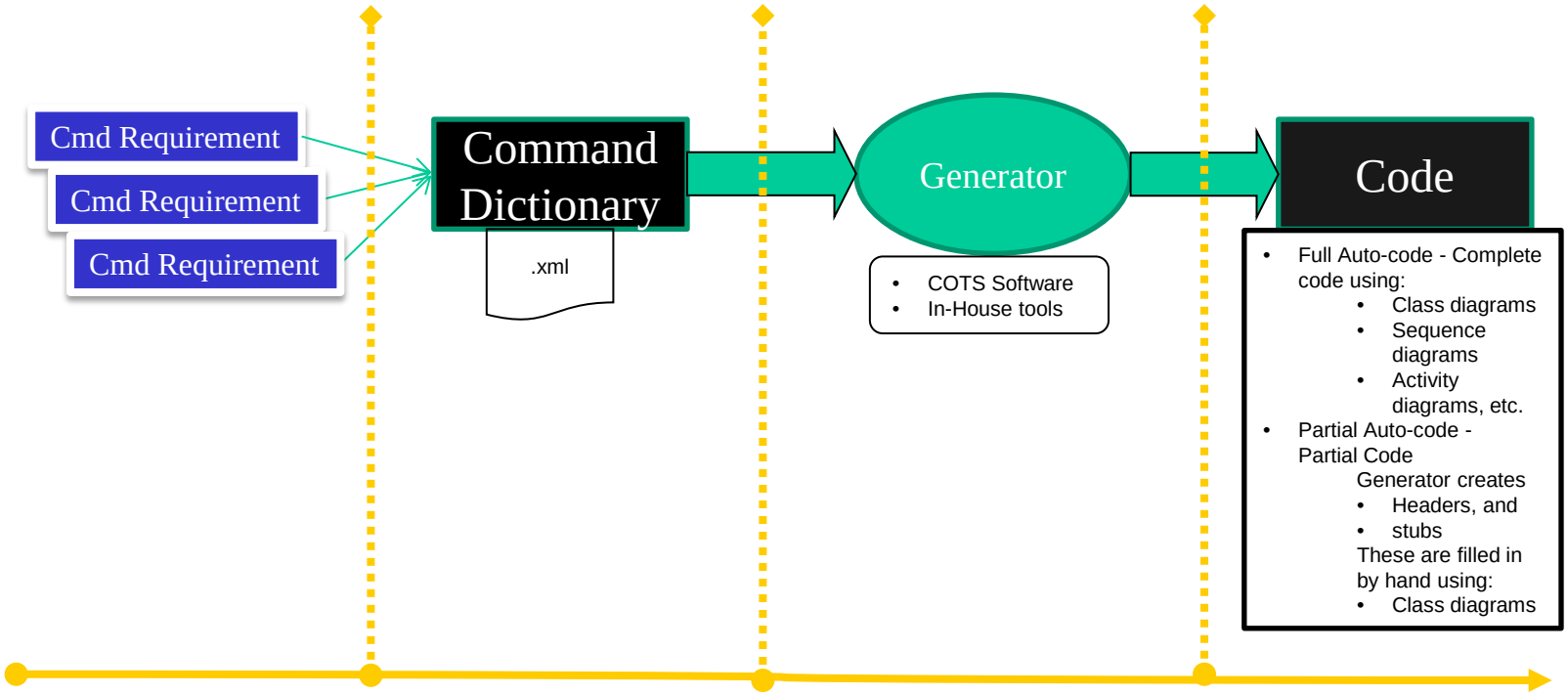
Automatic-Code Generation Flow



IV&V Facility

How current IV&V methods treat Auto-Code

Automatic-Code Generation Flow from Requirements



Areas of IV&V Focus

- What/Where should IV&V focus be?
  - Analyze the Input?
  - Analyze the Generator?
  - Analyze the Output Code?
  - IV&V artifacts from all three?

Areas of IV&V Focus

- Analyze the Input?
  - Models – design blueprints based on knowledge of systems and requirements
    - Matlab
    - Simulink
    - Python
    - MatrixX
    - Rhapsody
    - Template-Based
  - Requirements

- Major Weaknesses:
  - Focus is only on what is modeled by the inputs
  - Abnormal behaviors not defined in inputs
  - Derived & Implicit Requirements are more likely to be missed
  - IV&V techniques for analyzing software models instead of documents are not very prevalent

Areas of IV&V Focus

- Analyze the Generator?
  - [1]Should/Can IV&V trust the Generator?
    - Is the generator a COTS product?
    - Has IV&V been performed on it?
    - Can “Certification” of Generator be substituted for IV&V?

[1] *Jacob.T.Cox@IVV.NASA.Gov, “Arguing with the Machine”*

Areas of IV&V Focus

- Analyze the Generator?
  - In IV&V contexts, the autocode generators mostly fall into the same category as COTS Software

- Weaknesses:
  - At this point, known IV&V weaknesses are exactly identical to those of COTS software (e.g., Operating Systems):
    - Cost
    - Having to rely on V&V performed by vendors of the COTS software
    - In any given system's software, some COTS features will not be required and subsequently, "Dormant Code" will always exist
    - IV&V techniques for COTS software are limited to "black-box" approaches
    - Inability to look at Code

Areas of IV&V Focus

- Analyze the Output?
  - Full Auto code
  - Partial Auto code
  - Target languages
    - C/C++, ADA

- General Issues:
  - Lack of code documentation – most critical issue noted so far
  - Resource Leaks are common
    - Memory
    - Mutex
    - Files
    - Garbage Collection not strongly enforced or implemented
  - Are there Target Processor specifics (e.g. Big vs. Little Endian compatibilities)? – Yet unknown.
  - Garbage Collection not strongly enforced or implemented

What we are seeing

- Strong subject matter expertise focus
  - Domains (e.g. GN&C)
- Issues unique to individual code generators
- Example Mission:
  - Fewer errors than Hand coded
  - Focus is on Design Analyses of Input Models
  - 75% of Code is auto generated
  - Static Code Analyses found zero Severity 1-3 issues

What we are seeing

- Another Mission:
  - IV&V opted not to look at the auto-generated code from MATRIXx -- it is “very messy and hard to read”.
  - Estimate is that the auto-generated code from MATRIXx is six to seven times as much more code compared to the equivalent hand generated ADA code.
    - Negative impact on Worst-Case Execution Times

Recommendations

Can the common issues be remedied by having a better-input model? How about focusing IV&V on the Output Code?

- We recommend:
  - Adding to IV&V, methods that address these weaknesses
  - Dynamic Analyses
    - Remedying some of the issues (e.g. memory & resource management + Execution Time impacts) observed suggests Dynamic Analyses
  - NASA IV&V Project Execution Plan (IPEP) with slight adaptations for Rules of Engagement with respect to IV&V of auto-generated code – straddles

Questions?