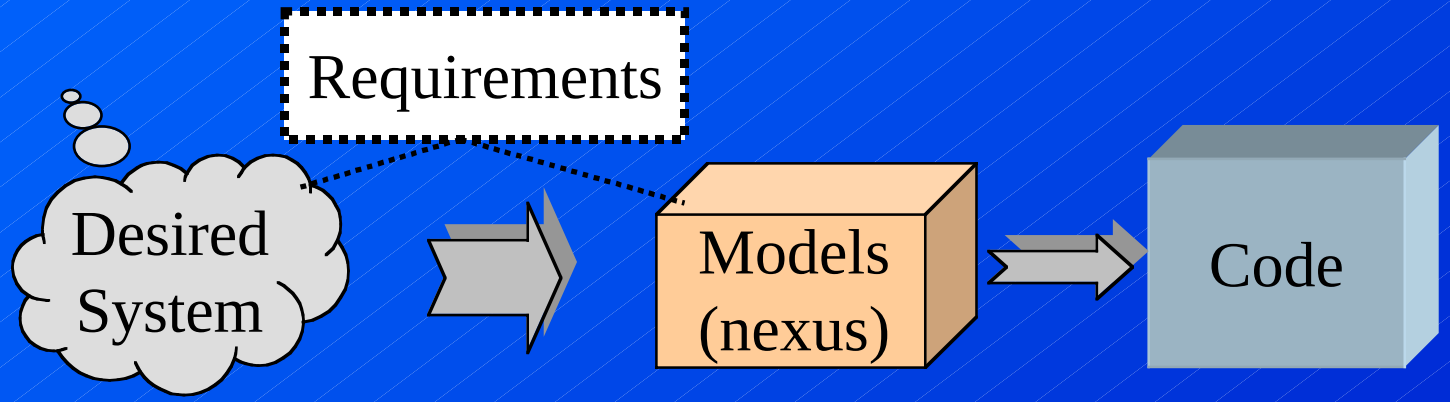
Dependability Challenges in Model Centered Software Development

David P. Gluch
Embry-Riddle Aeronautical University
&
Visiting Scientist
Software Engineering Institute
Carnegie Mellon University

Context Model Centered Development



Contrast:

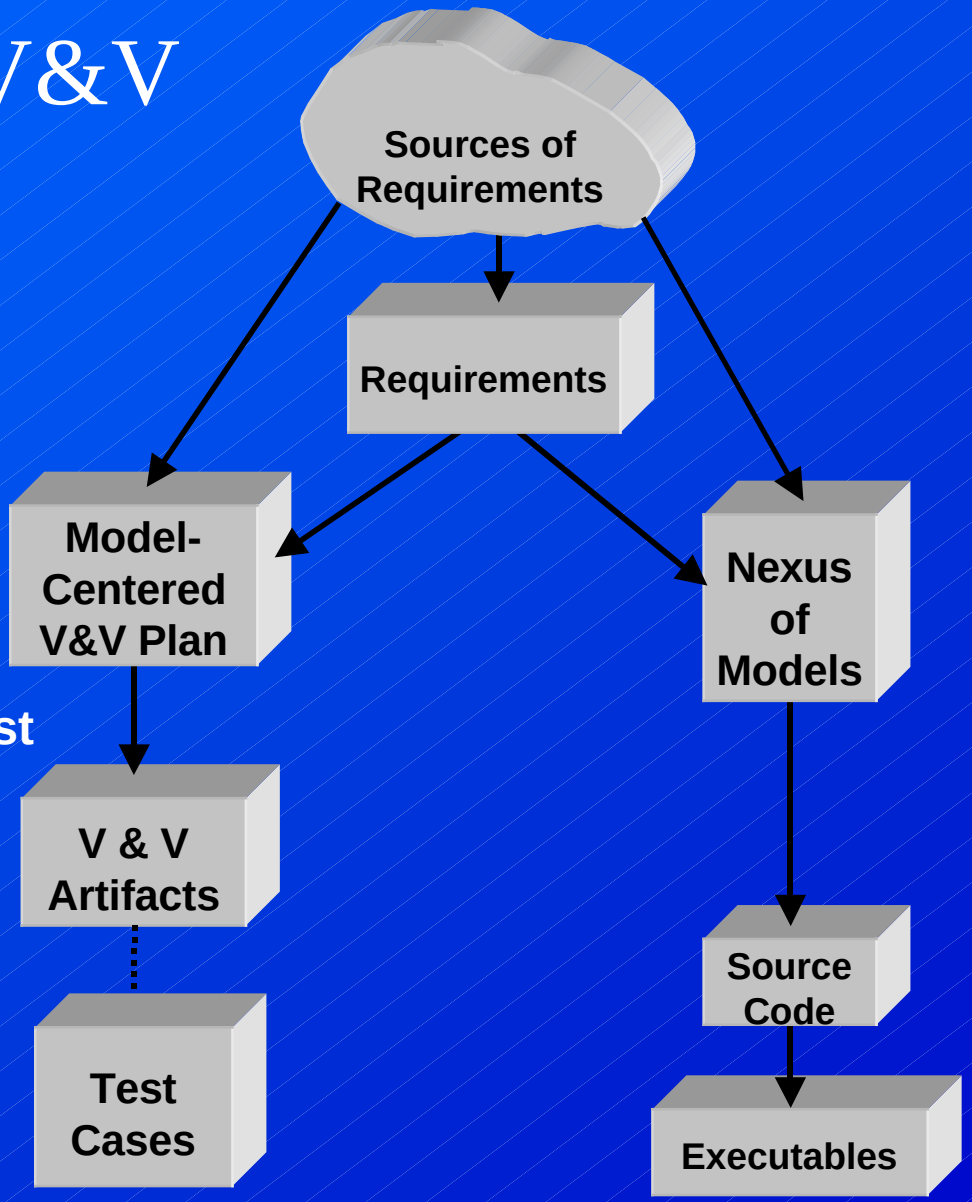
- **Sequence of phases**
- **Requirements**
- **Architecture**
- **Detailed Design**
- **Coding**

Attributes

- **Minimal Textual Requirements**
- **No Sequence of Documents**
- **Multiple Modeling Views**
- **Increasingly Formal Models**
- **Conformance to Standards (e.g. UML)**
- **Integrated Tool Support**
- **Automated Code Generation**

Concurrent V&V

- Variable levels of autonomy
  - Integrated to independent
- Formal Models
- Model Analysis
- Coordinated Analysis and Test



Challenges : Analysis of Models

Assess and establish viability in real world mission and safety critical software development

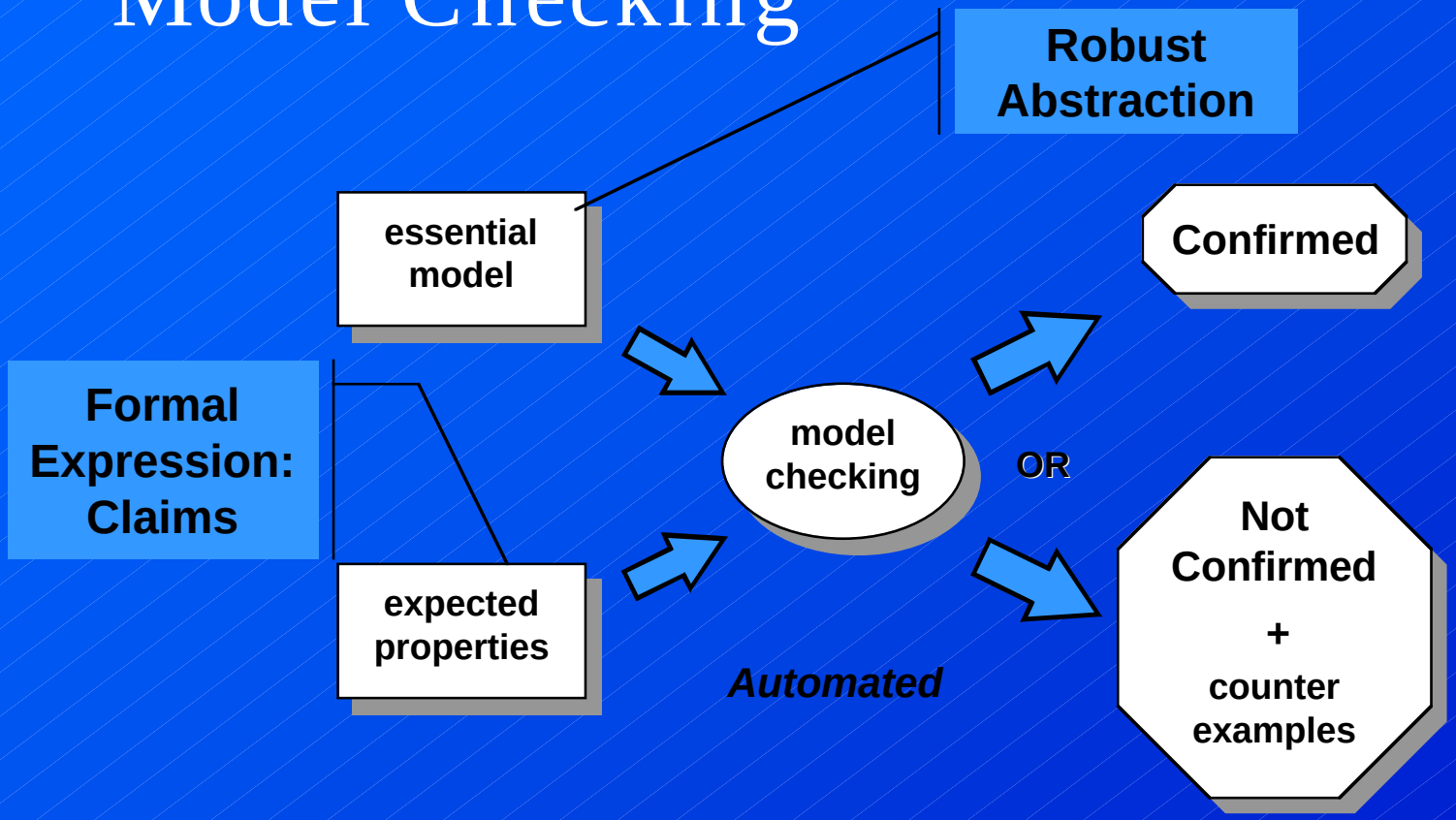
Foundational Research

- **Enhanced model checking approaches**

Transition - Facilitate Adoption

- **Practices**
- **Education**
  - **Change software engineering “culture” and “thinking” (analyze and design)**
- **Commercialization**
  - **Deformalize formalism**
  - **Standardized approaches (e.g. UML, OCL)**

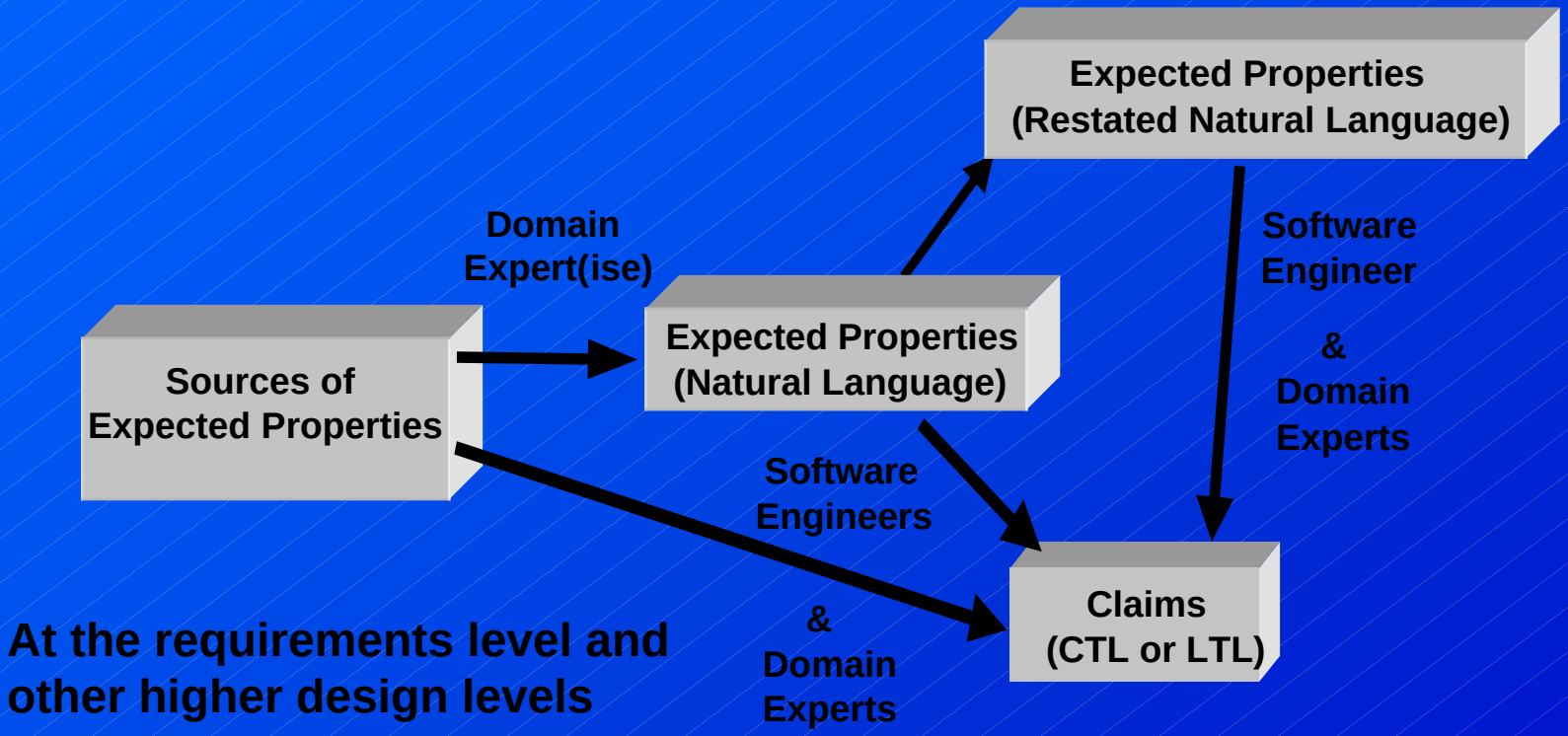
Model Checking



Transition (and research) Challenges:

- **Abstraction**
- **Generating expected properties (queries)**
- **De-formalizing Claims**

Expected Properties and Claims

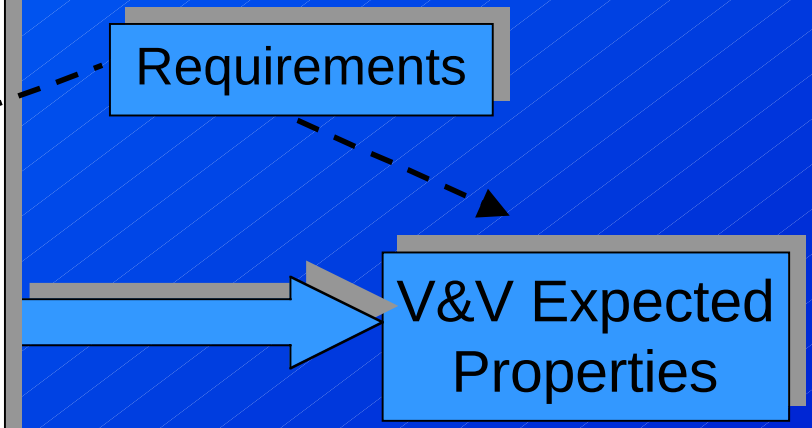


Sources

Application Domain
- Users
- Customers
- Operations Personnel
- Maintenance Personnel
- Domain Experts
- *Requirements*

Technology Domain
- Technology Experts
- Technology Standards

Development Methodology
- Development Experts
- MBV Experts
- Quality Assurance Personnel
- Quality Standards
- Standard Practices
- Engineering Standards
- Development Technique
- Intra and Inter-model



At the requirements level and other higher design levels

Strategies and Heuristics for generating expected properties

De-Formalizing Claims

Consider Computation Tree Logic Expressions

AGAF (agitator = engaged)

AG (temperature = high -> agitator = engaged)

AG ((EX engine = ignition) -> safety-lock = released)

! EF(AG (state = idle))

Template Classifications - Taxonomy

Occurrence

- Basic reachability
- Transitionability
- Global reachability
- Infinite occurrence
- Qualified occurrence
- Co-Occurrence
- Permanent occurrence
- Error free execution
- Mutual exclusion

Cause & Effect

- Simple cause – effect
- Permanent cause – effect
- Cause – scoped effect
- Cause – chained effects
- Immediate precondition
- Chained causes – effect

Non-progress

- Deadlock

  Starvation

Template: Qualified Occurrence

Predicate 1 is true at least until the first occurrence of Predicate 2 and Predicate 2 will eventually become true.

```
CTL: A [ Predicate 1 U Predicate 2 ]
LTL: Predicate 1 U Predicate 2
```

Note that Predicate 1 does not need to change to false when Predicate 2 occurs. It may continue to be true.

Examples and known uses:
Sometimes a condition must hold from the initialization of the system until something happens. For example:

```
A [ Ejection = disabled U Plane has taken off ]
```

Template: “Cause and Effect"

If the pilot presses the ejection button, the seat will be ejected.

```
AG (
        Ejection_button = pressed  ->
AF (Seat = ejected)
       )
```

Note that the seat may be ejected any number of cycles after the ejection button being pressed.

Related claims and templates:

For an immediate effect (in the next state), AX can be used instead of AF.

For a possible but not guaranteed effect, use EF instead of AF.

Challenges – Summary

Effectively Integrate Formalism

Robust model analysis and checking strategies

Lack of Commercial Tools

Correlate model analysis with testing

Need both domain and technical expertise