MODEL LEVEL DESIGN PATTERN INSTANCE DETECTION USING ANSWER SET PROGRAMMING

MISE 2016 –
Session: MDE technologies and Model Quality

Gaurab Luitel, ***Dr. Matthew Stephan,*** & Dr. Daniela Inclezan
Miami University, Oxford, Ohio

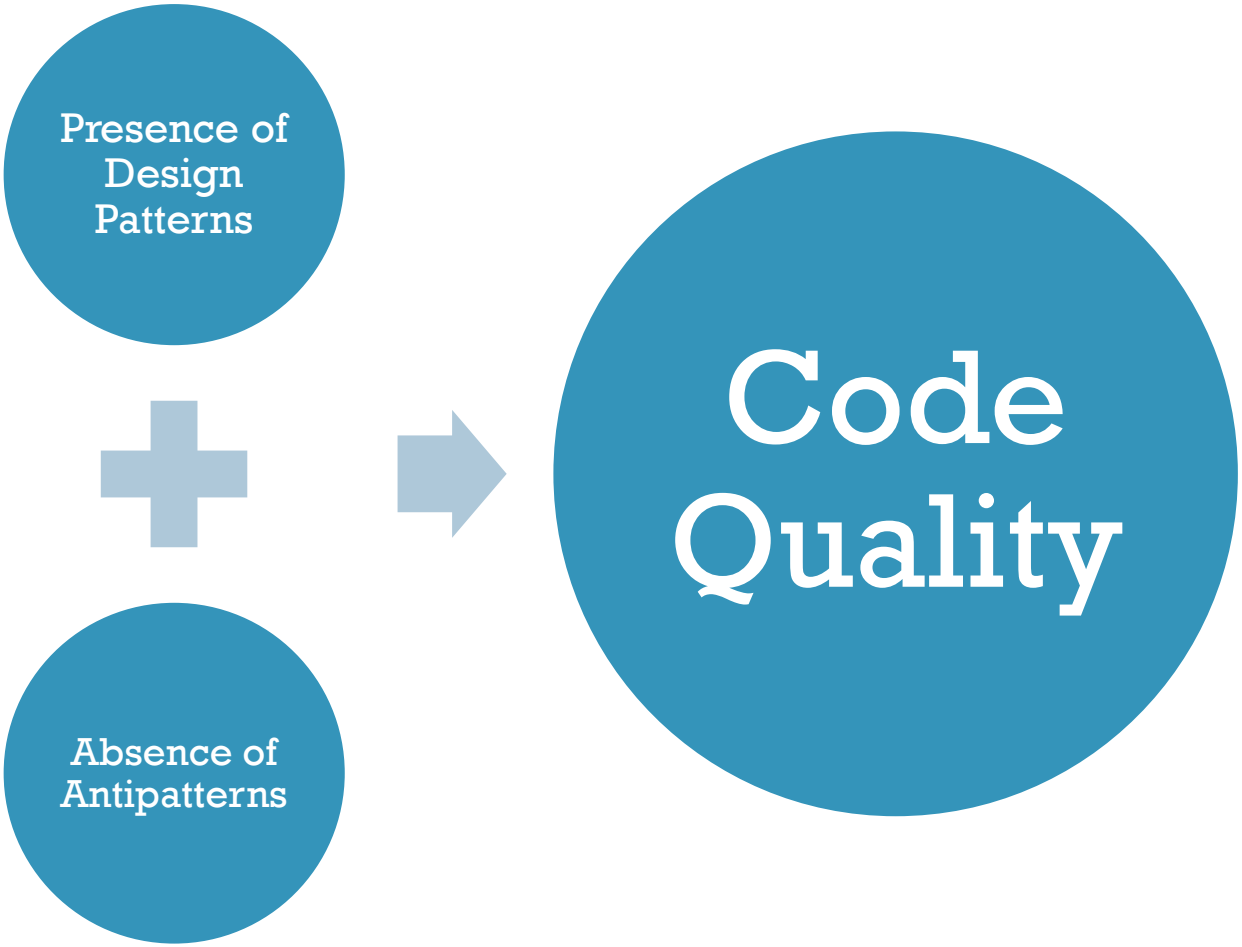
INTRODUCTION

- Important MDE Problem: Quality of Modeling Artifacts
  - How do we assess quality of our artifacts in MDE?
    - Metrics
  - Needs Improvement
    - Quality Assurance of Traditional Software >> QA MDE
- Ideal world
  - Automated analysis
    - Large model sets
    - Incomplete models
  - Determine desired and undesired properties

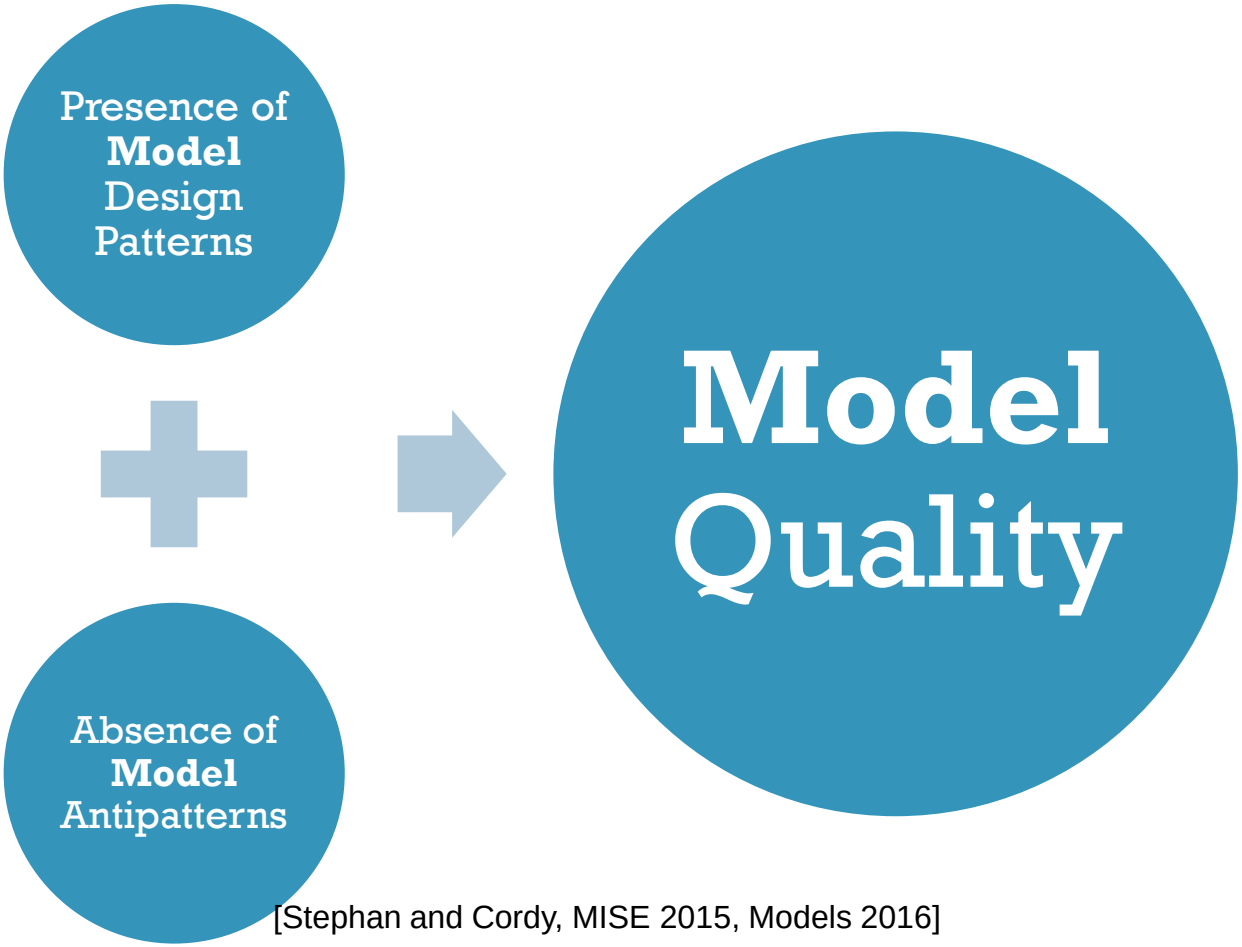
Patterns as a Measure for Quality

- One established approach to assess software quality

  (Houston, 2001; Van Emden, 2002; More)



SO, WHY NOT?



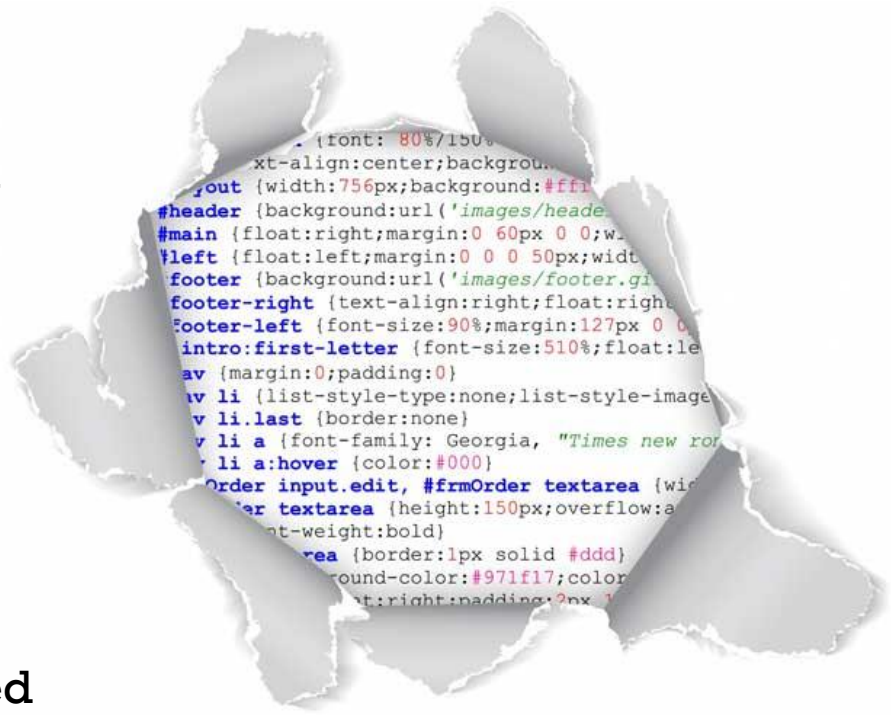
[Stephan and Cordy, MISE 2015, Models 2016]

DISCUSSION POINTS FOR LATER

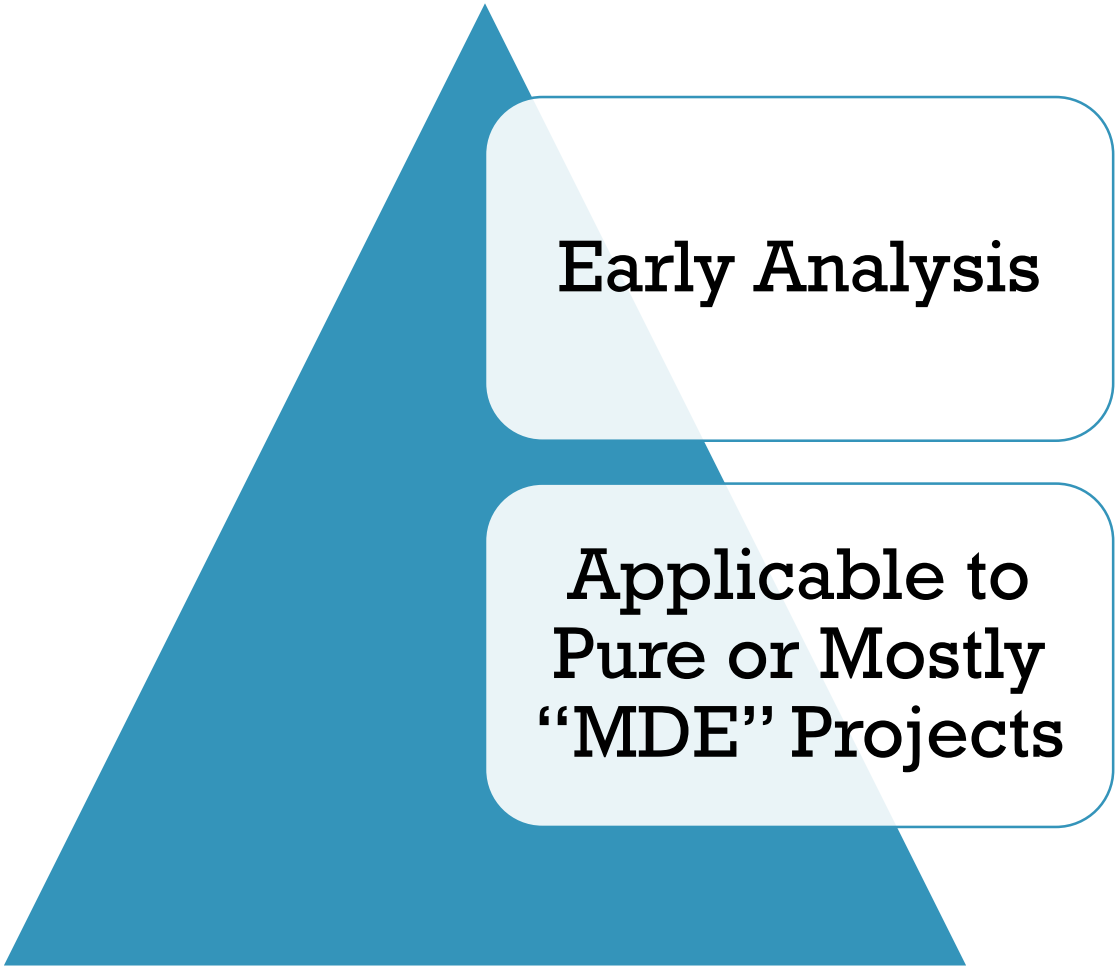
- Patterns as a measure of quality?
  - Despite validated work for code and models, do you believe?
  - Philosophically, does it make sense?
  - Why/why not?
- Keynote:
  - Patterns viewed as "correct rewrites"?
    - Equivalent but better?
  - Structural and behavioral identies

PATTERN INSTANCE DETECTION

- Majority of approaches analyze source code
  - Wait until code is generated from models
  - E.g., extract out metadata from C++ source, compare to Prolog Rules
- Reverse Engineer code into other forms
  - Code -> Matrix
  - Code -> Models
- But we want to provide QA on the models themselves!
  - Many patterns are already presented and abstracted in model form!



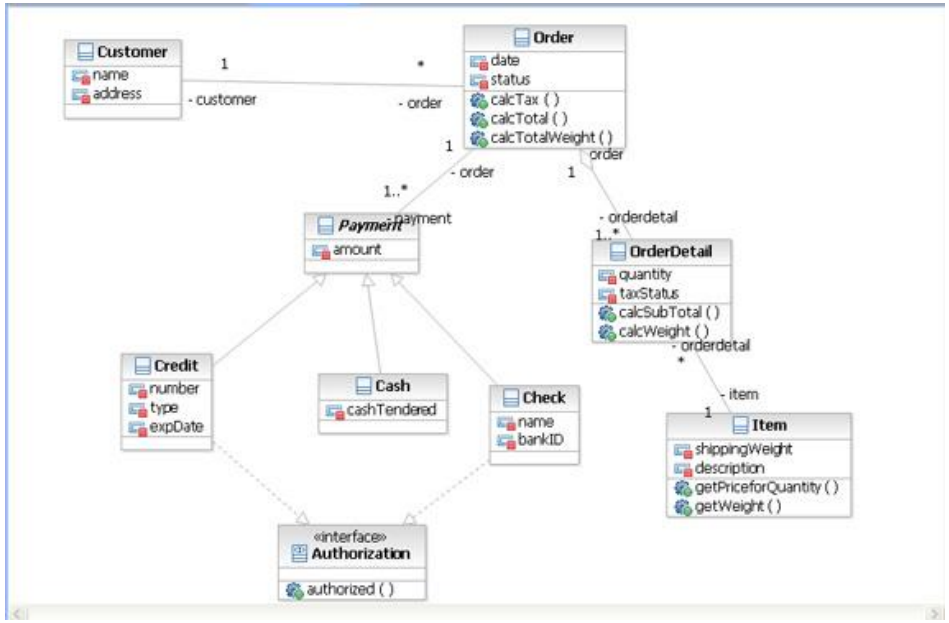
BENEFITS OF MODEL LEVEL DETECTION



[Stephan and Cordy, MISE 2015 and Models 2016]

PROBLEMS WITH EXISTING MODEL LEVEL PATTERN DETECTION

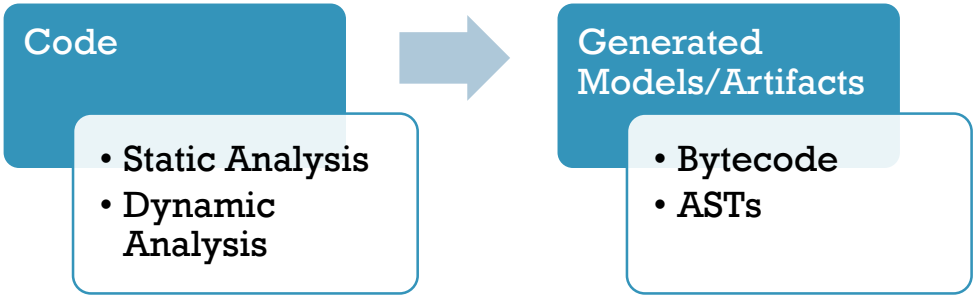
- Most work focuses solely on structure, disregards behavior
- Problem:
  - Structural information alone is not always sufficient for software pattern detection[1]
  - False positives/Low precision



[1] I. Bayley and H. Zhu. Formal specification of the variants and behavioral features of design patterns. Journal of Systems and Software, 83(2):209-221, 2010.

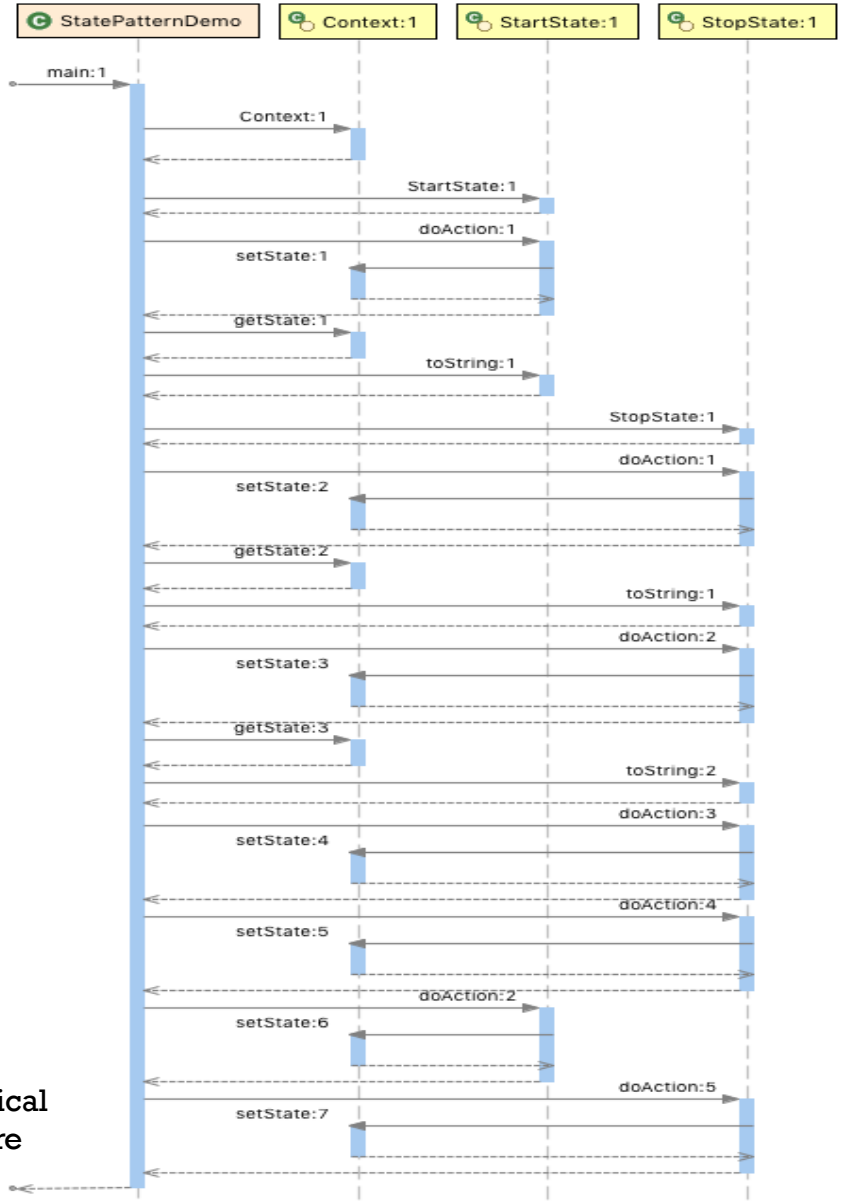
PROBLEMS WITH EXISTING MODEL LEVEL PATTERN DETECTION

- Existing work that considers behavioral aspects requires structural models AND source code
- Problem:
  - No longer "Model Level"
  - Precludes
    - Early analysis
    - Pure MDE Environment
- Examples
  - Code -> UML -> Rules
  - Dynamic Code Analysis
    - Bytecode
    - ASG



WHICH BEHAVIORAL MODELS TO USE?

- An existing approach uses Collaboration Diagrams
- We choose Sequence Diagrams. Why?
  1. Sequence diagrams are more commonly used in industry[2]
  2. More helpful since more concerned with temporal aspects
  3. Already been defined explicitly for many patterns in the literature



[2] J. Hutchinson, J. Whittle, M. Rouncefield, and S. Kristoffersen. Empirical assessment of MDE in industry. In International Conference on Software Engineering, pages 471-480. 2011.

DISCUSSION POINTS TO CONSIDER LATER

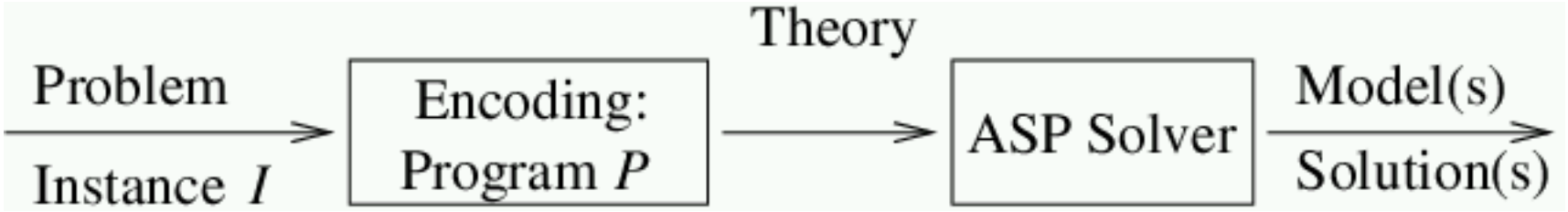
- Support the decision to use Sequence Diagrams?
  - Why or Why Not?
- Thoughts on the necessity of behavioral features/aspects explicated in pattern definitions.

REASONING ABOUT MODELS

- RQ: Given the need to perform analysis on both structural and behavioral models, what can and should we use to reason/search for pattern instances?

PROPOSITION: ANSWER SET PROGRAMMING

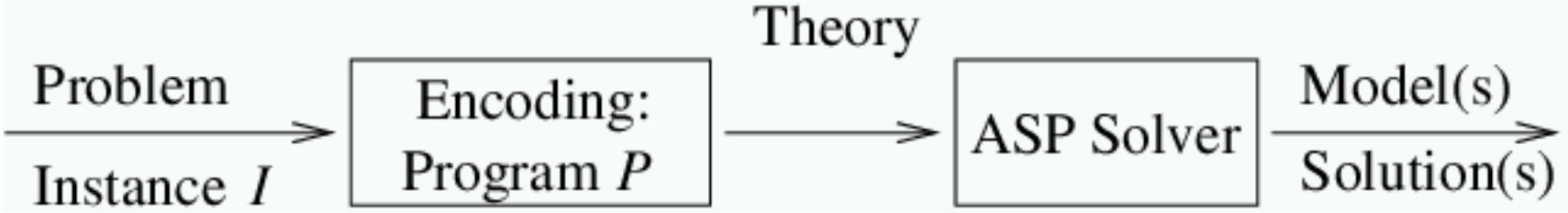
- Declarative form of logic programming
- Specifically geared towards complex search problems
- Prolog syntax, but underlying computation quite different
  - Stable logic programming model
  - Uses answer sets



http://www.kr.tuwien.ac.at/research/projects/WASP/asp-sep.gif

PROPOSITION: ANSWER SET PROGRAMMING

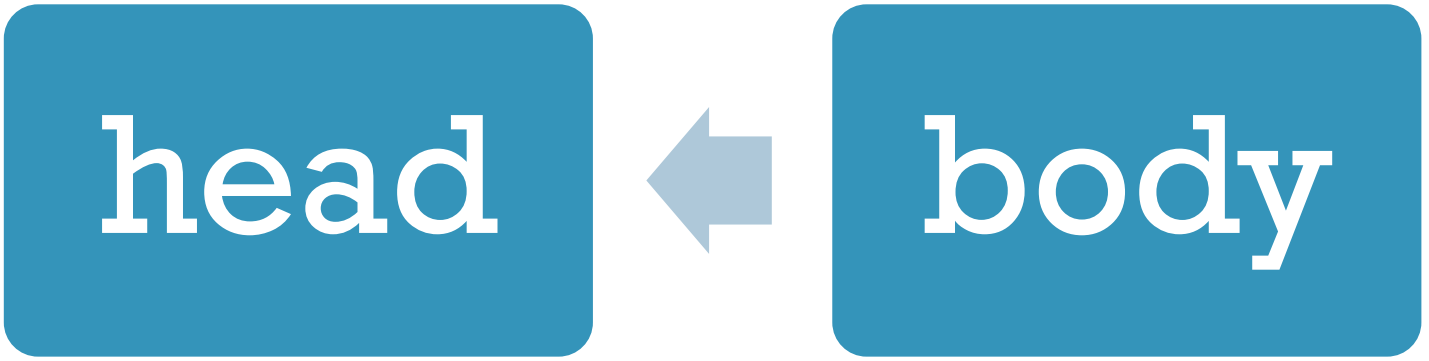
- Non-monotonic = New information can cause "true" predicates to be retracted
- Allows
  - Natural ASP representations of natural language statements
  - Exceptions through the use strong negation and default negation
- Especially suitable for representing qualitative knowledge
  - E.G., the knowledge we plan on encoding in
    - Class and Sequence diagrams
    - Default statements and their exceptions
    - Dynamic domains: change is triggered by actions
    - Uncertainty



http://www.kr.tuwien.ac.at/research/projects/WASP/asp-sep.gif

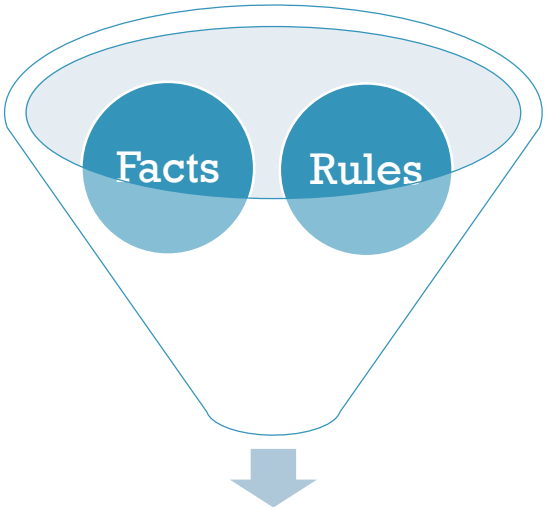
HOW DOES ASP WORK?

- Key is Answer Sets
  - "Believe ***head*** if you believe ***body***"
  - head is a literal (atom or its negation in FOL)
  - body is a set of literals
    - possible preceded by not = "there is no reason to believe"
  - Atoms and literals: Express properties of domain objects and relationships between objects



HOW DOES ASP WORK?

- Key is Answer Sets
  - Consist of literals that are believed to hold
  - 1 program can have multiple answer sets
    - Each answer set = belief set
- Answer sets are computed by inference systems called solvers
- Syntax consists of rules and facts



Search Results/
Stable Models

ASP VERSUS RELATED FOL APPROACH

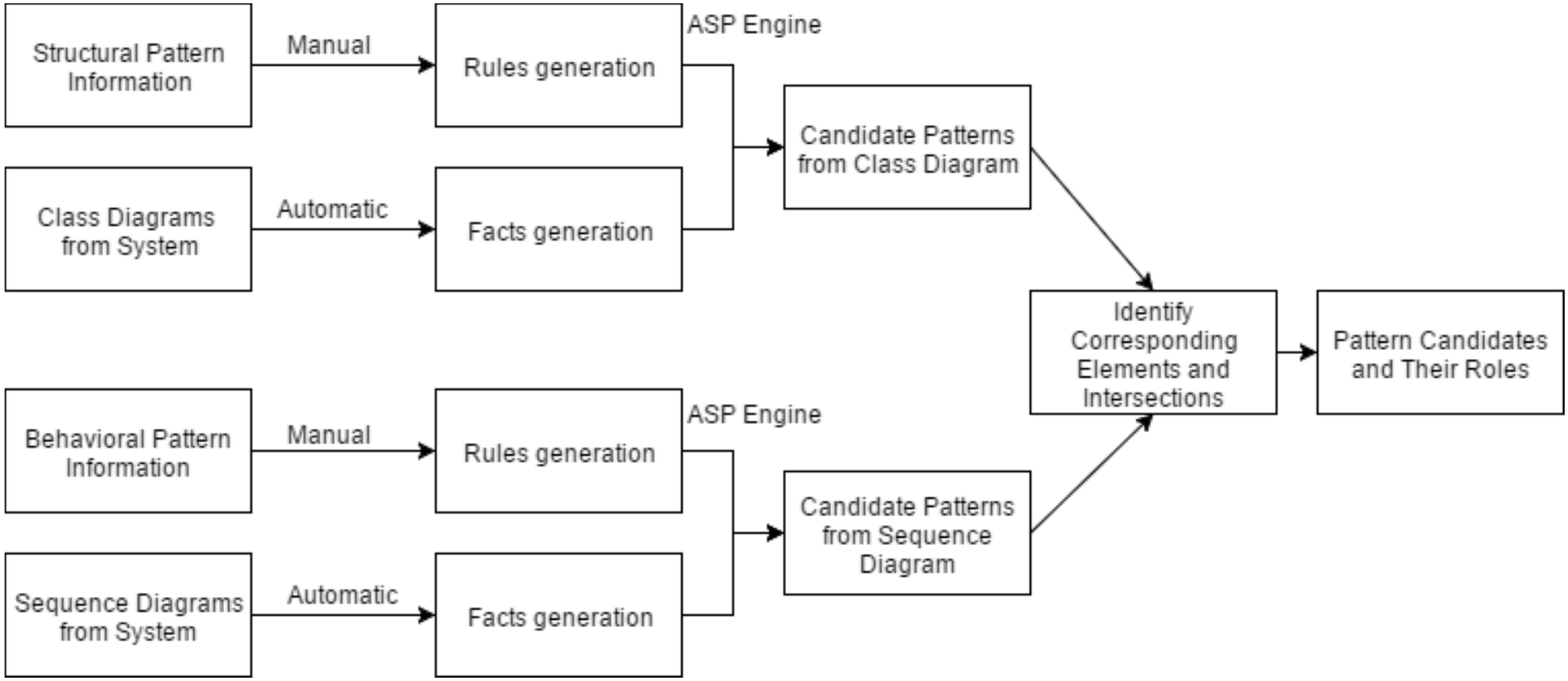
SPASS

1. SPASS can run forever/time out without any results
   - Their experiments present multiple time outs
2. Moves forward to what is being proven
3. Limitation on number of rules

ASP

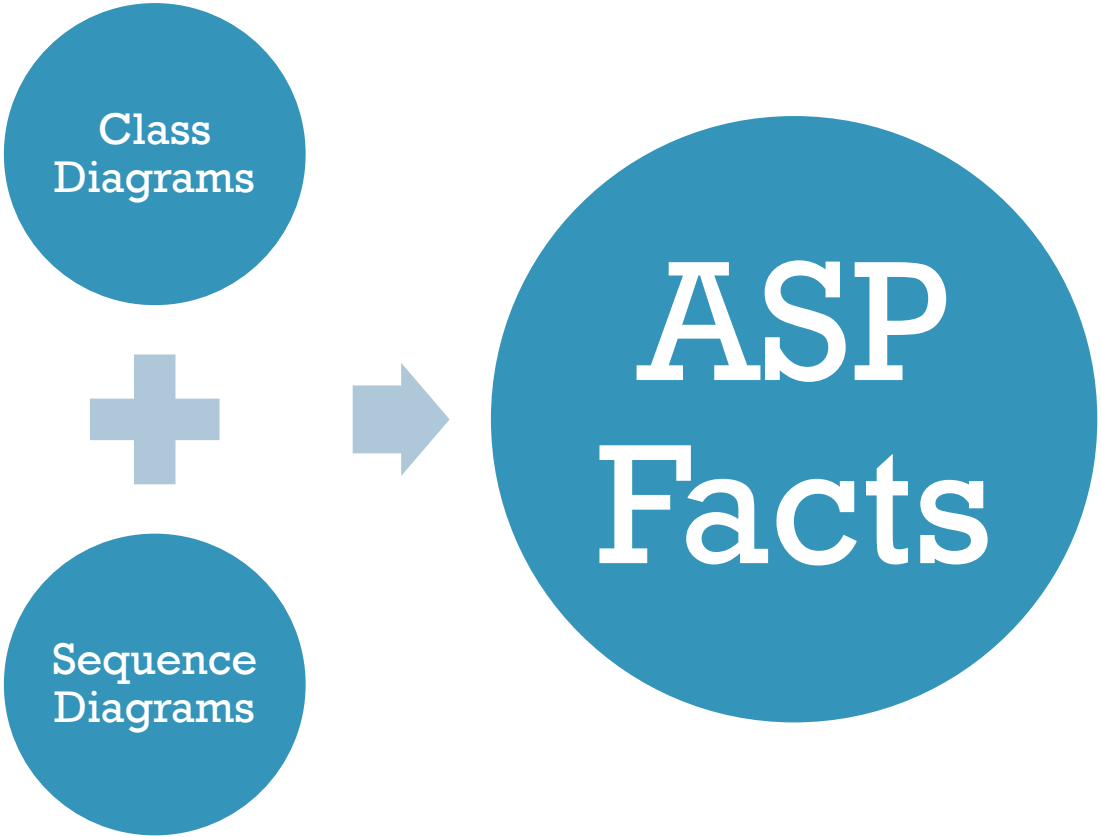
1. ASP always terminates, in principle
   - Tailored to these type of search problems
2. Has no notion of forward or backward inference
   - Multiple answer sets!
   - Example in paper
3. ASP does not have a limit on rules it can handle
4. Weakness: Computation < when specific cycles like default negation or function symbols -> infinite inputs
   - Non issue in class and sequence diagrams

APPLICATION OF ASP TO MODEL LEVEL PATTERN DETECTION



SYSTEM FACTS

- Abstract: Take in class and sequence diagrams
  - Represent them as ASP Facts



SYSTEM FACTS

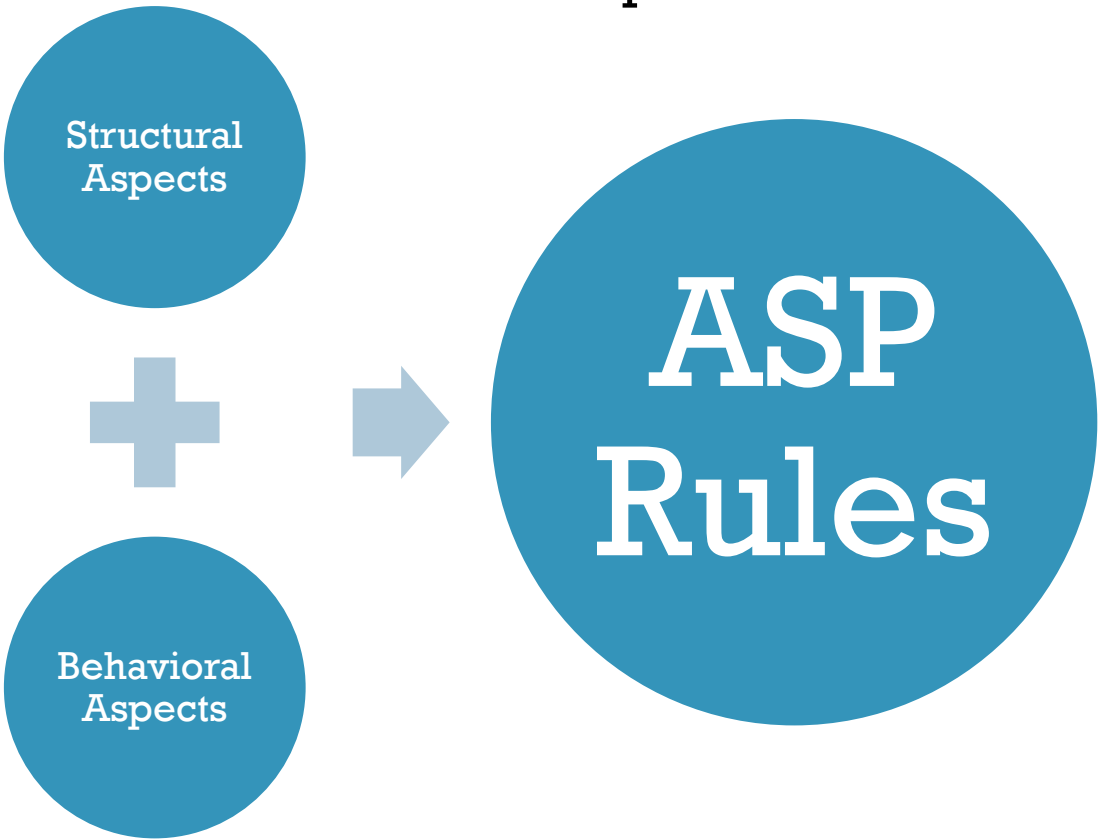
- Prototype: Require models in XMI form
  - Automate transformation from XMI -> ASP Facts
  - XMI = Prevalent
  - Can produce example/test UML and export to XMI
    - StarUML Tool

DISCUSSION POINTS TO CONSIDER LATER

- System Facts – Feasible?
  - We leverage existing work transforming XMI to first order logic (LAMBDES-DP)
  - Still, we can discuss your skepticism.
  - Keynote:
    Equivalent model sets: same facts?

PATTERN RULES

- Manual process (for now)
  - Acceptable since rule generation is rarely occurring task
- Encode structural and behavioral patterns into ASP rules



QUICK EXAMPLE – STATE PATTERN (DETAILS IN PAPER)

- Structure alone is insufficient (can lead to low recall and precision)
- Sample established requirements [1]
  - 2) Requests are the operations of the context
  - 3) Handlers are the operations of the state
  - 5) All handlers must be abstract (not concrete)

$$class(context, [state], [context, getState, setState], no).$$
$$operation(state, doAction, void, [context], yes, no, no).$$
$$operation(startState, doAction, void, [context], no, no, no).$$
$$operation(context, setState, void, [startState], no, no, no).$$

$$message(13, statePatternDemo, startState, doAction).$$
$$message(131, startState, context, setState).$$
$$message(16, statePatternDemo, stopState, doAction).$$
$$message(161, stopState, context, setState.$$

$$2) isRequest(Op) :- class(C, \_, OpList, \_), isContext(C),$$
$$\#member(Op, OpList), operation(\_, Op, \_, \_, \_, \_, \_).$$
$$3) isHandler(Op, C) :- class(C, \_, OpList, \_),$$
$$isState(C),$$
$$member(Op, OpList),$$
$$operation(\_, Op, \_, \_, \_, \_, \_).$$
$$5) isHandler(Op, C) :- operation(C, Op, \_, \_, no, \_, \_).$$

DISCUSSION POINTS TO CONSIDER LATER

- Automate Pattern Rule Development?
  - Pattern inference? Interest research topic!
  - Union Pattern mining work with ASP rule transformation
- Validation?
  - Develop patterns incrementally
  - Test on many variations (Mutation Analysis?)
  - Refinement Infinite process!
- Recall and Precision
  - Extra slide, if interested
  - Expect
    - > Precision
    - Ideally ~ Same Recall (with tuning)

STATUS AND IMMEDIATE PLANS

- Written ASP rules representing a variety of design patterns
- Manual fact generation
  - Automation is key!
    - Currently working on XMI -> ASP Facts (Goal: Summer 2016)
- Tested on toy (relatively small and contrived) systems
  - Positively identified pattern instances and pattern roles
- Validation
  - Compare our results to existing approaches
  - We can also reverse engineer diagrams from source code
    - Solely for testing/validation purposes of the detection algorithm, not for SE validation
  - Inject and mutate some pattern instances in larger systems

CONCLUSIONS

Technique to detect model patterns using ASP

- Analysis directly on the models in lieu of source code
- Use as a measure of model quality

Represent structural and behavioral pattern aspects using FOL

- Structural and behavioral pattern aspects as ASP rules
- Systems represented as ASP Facts

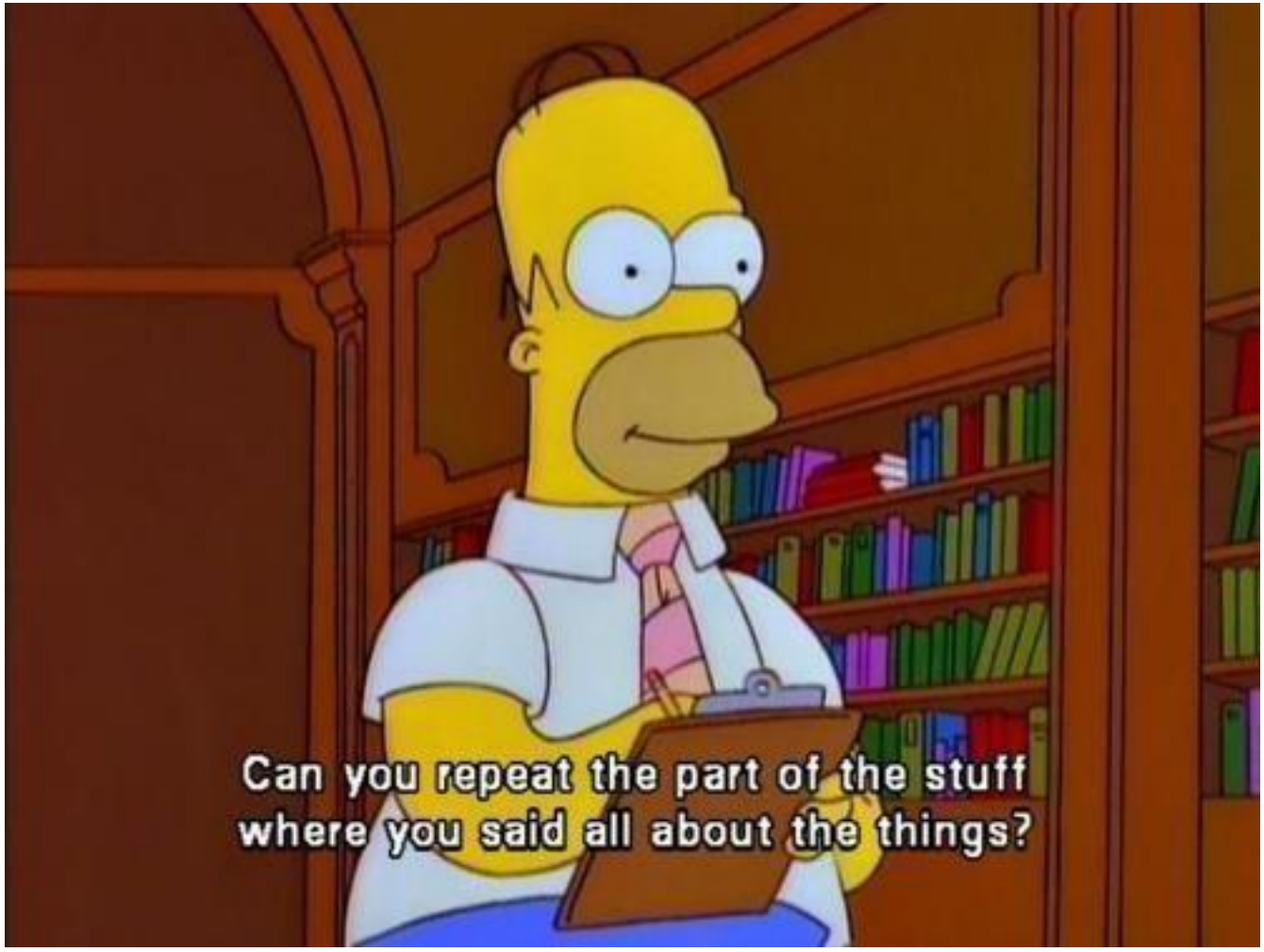
Plan on identifying instances efficiently and accurately

- > Methods that consider structure only
- ASP has advantages over other FOL techniques for this purpose

Interesting research aspects and potential milestones

- Formalize patterns as ASP rules
- Automating transformation of system to facts
- Validation through comparison of both model-based and code-based techniques
- Using ASP in this manner should help facilitate MISE by improving model analysis and evaluation

QUESTIONS AND DISCUSSIONS?



EXTRA: RECALL AND PRECISION

- Expect >> Precision
  - More will be correct:
    - Increasing Information and increasing requirements!
- Ideally ~ Same Recall
  - Challenge
  - Matter of tuning specification pattern rules
- Our goal is to improve false-positive rate of SPASS approach
  - They have ~ false positive rate attributed to timeouts
  - Precision ~ Same because we apply and extend their formalisms
- Source code
  - > Precision
    - Understandable since code is more detailed
  - Recall ?
    - Code can yield noise since more details, so models can be better
    - Expect comparable recall

EXTRA: PRESENTATION OF CANDIDATES

- Simplest approach = Text Representation
  - XML?
    - <pattern > tag
    - <element tag> = UUID?
- Long term goal = Graphical Viewer
  - Highlighting
  - Labeling roles / Shading
  - Possible through UML UUIDs
- Leverage existing tools that visualize structural and behavioral analysis information