Architecture Synthesis and Dependability Evaluation of System Models

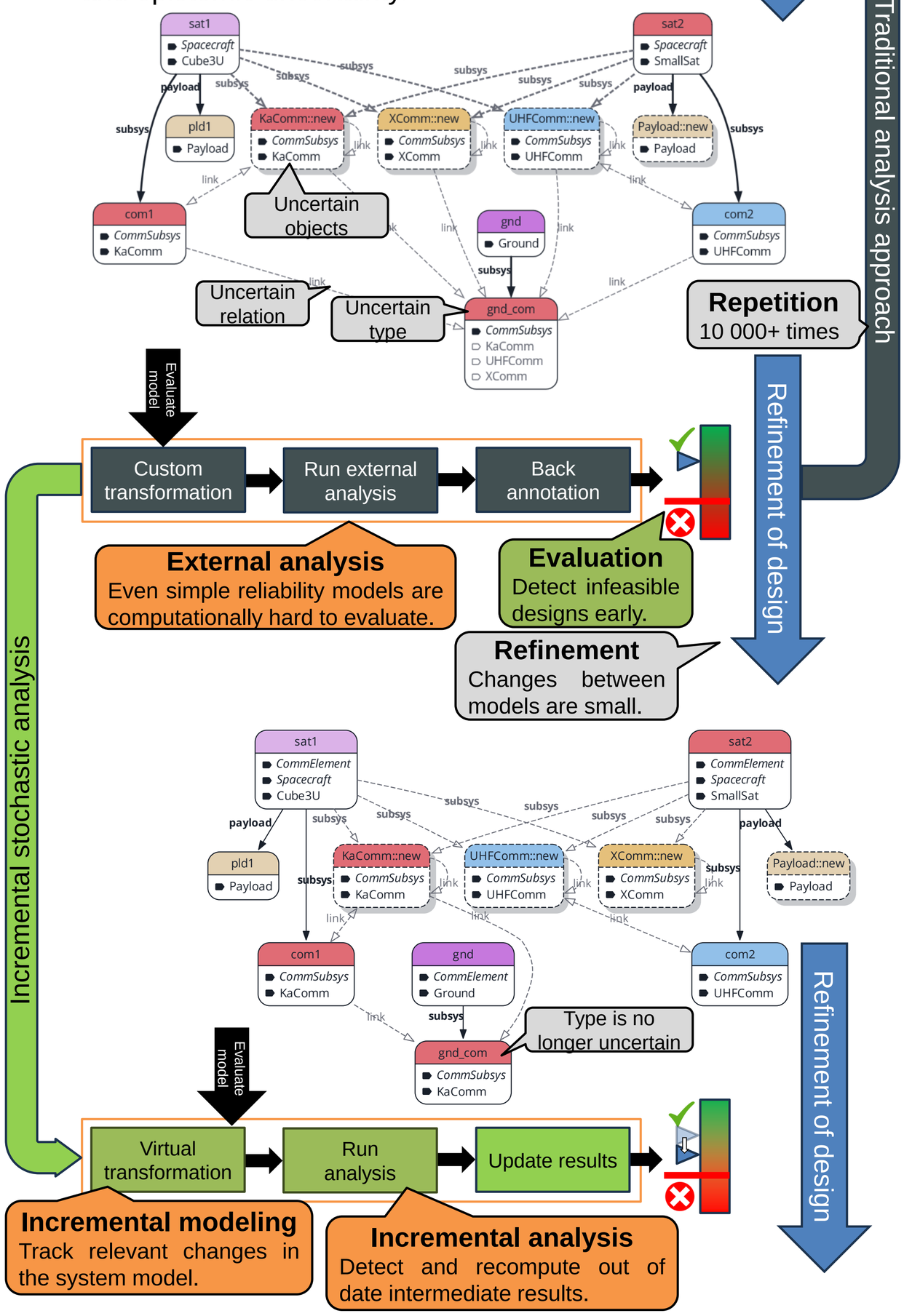
Máté Földiák
Dept. of Computer and Information Science
Supervisors: Dániel Varró, Lena Buffoni

Motivation

Safety and reliability play a crucial role in the operation of *critical cyber-physical systems*, such as trains, aircrafts or satellites. Selecting an optimal system architecture early, is imperative to satisfy such constraints while having an efficient design process, yet existing design space exploration (DSE) tools do not include sufficient support for probabilistic requirements, like minimum availability. My goal is to fill in this gap by developing a method, that supports efficient probabilistic analysis and reasoning in DSE.

From requirements to early system design

System architecture design relies on iteratively refining a candidate system architecture of an *arbitrary domain* by applying small mutations. Correctness is checked every iteration to guide the process towards an optimal, or close to optimal solution. However, stochastic analysis tool are unable to efficiently handle incremental changes and epistemic uncertainty.



Probabilistic Analysis with Graph Queries

We propose **probabilistic graph queries**, a formalism that extends the high-level VIATRA Query Language [2] with probabilistic semantics for precise analysis. It uses *lightweight language extensions* to insert probabilistic interpretation into queries, highlighted in grey bellow, and its semantics are in line with the semantics of regular graph queries. Furthermore, it supports *incremental analysis* through the VIATRA Query Engine and stochastic decision diagrams.

Language: basic events
Input from external sources that describe known reliability related properties of the component. In this case, availability.

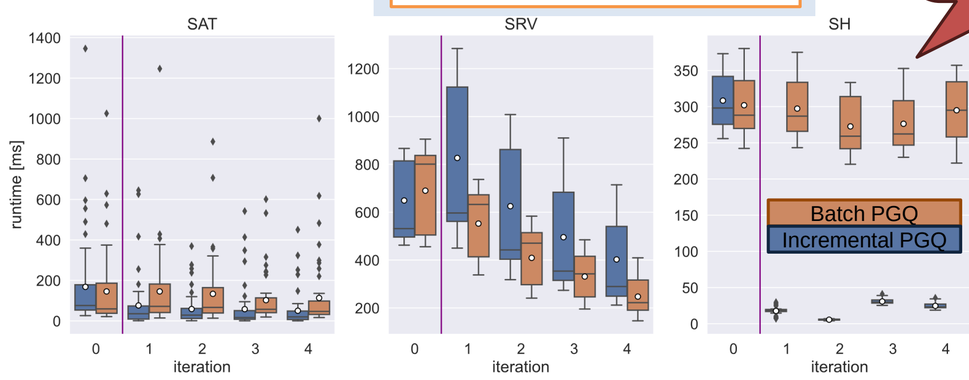
Example
Each KaComm type hardware component on a spacecraft is *operational* with a probability of ~90.5%.

Language: quantification
The user can quantify event probabilities and derive custom metrics to check extra-functional consistency.

```
pattern link(src: CommSubsys, trg: CommSubsys){
        CommSubsys.target(src, trg);
} or {
        CommSubsys.fallback(src, trg);
}

@BasicEvent(probability=0.90483741803596)
pattern operationalKA(obj: EObject){
        Spacecraft.subsys(_, obj);
        KaComm(obj);
} //…

@CompoundEvent
pattern operational(obj: EObject){
        find operationalKA(obj);
} or // ...

@CompoundEvent
pattern ready(comm: CommSubsys){
        GroundStation.subsys(_, comm);
} or {
        Spacecraft.subsys(sat, comm);
        find online(sat);
        find operational(comm);
}

@CompoundEvent
pattern online(sat: Spacecraft){
        Spacecraft.subsys(sat, comm);
        find operational(sat);
        find operational(comm);
        find link(comm, trg);
        find ready(trg);
}

@CompoundEvent
pattern available(pld: Payload){
        Spacecraft.payload(sat, pld);
        find online(sat);
}

@Weight(lhsname="cvr", class="Performability",
        function="calculate")
pattern coverage(cvr: java Double){
        cvr == Weight find available(_);
}

pattern good(){
        find coverage(quality);
        check(quality >= 0.3157);
}

class Performability {
        public static double calculate(int count){…}
}
```

Language: compound event
Indicators, whether a higher function (or service) satisfy a condition, dependent on other imperfect components and functions.

Example
A CommSubsys is *ready* (to receive data), if it is equipped to the ground station, or if the containing satellite is *online*.



Result
Can reduce analysis time, depending on domain complexity and size of change.

Extending the expressiveness by introducing more event types [1], integrating it to Refinery, and developing the necessary formalism for handling 4-valued partial models as inputs is work in progress.

References

[1] Probabilistic Graph Queries for Design Space Exploration Under Uncertainty
Máté Földiák,
2024, MODELS 24 Doctoral Symposium

[2] Road to a reactive and incremental model transformation platform: three generations of the VIATRA framework
Dániel Varró, Gábor Bergmann, Ábel Hegedűs, Ákos Horváth, et al.
2016, International Journal on Software and Systems Modeling (SoSyM)

[3] Refinery: Graph Solver as a Service
Kristóf Marussy, Attila Ficsor, Oszkár Semeráth, and Dániel Varró
2024, ICSE: Tool Demonstration Track