From Behaviour-Driven Development (BDD) Scenarios to Formally Verifiable Behavioural Models via Dynamic Condition Response (DCR) Graphs

Extended Abstract

Part of the AI4SE Grand Solutions research project funded by Innovation Fund Denmark

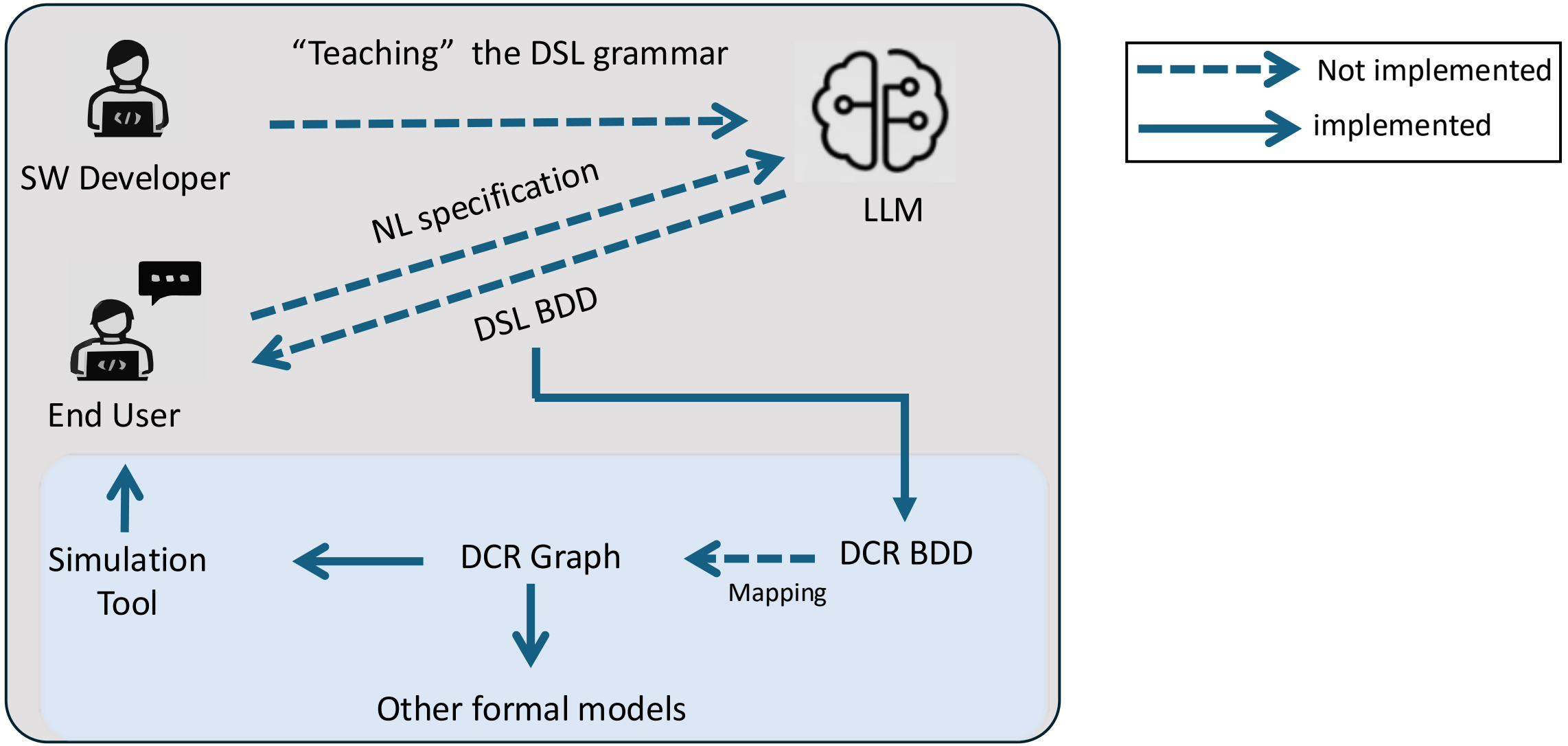
Xinyuan Tu[1], Thomas Hildebrandt[1], Thiago Rocha Silva[2]

University of Copenhagen[1],
Southern Denmark University[2]

Agenda

1. Overview
2. DCR graph-related works
3. Works in progress
4. Future works

Overview



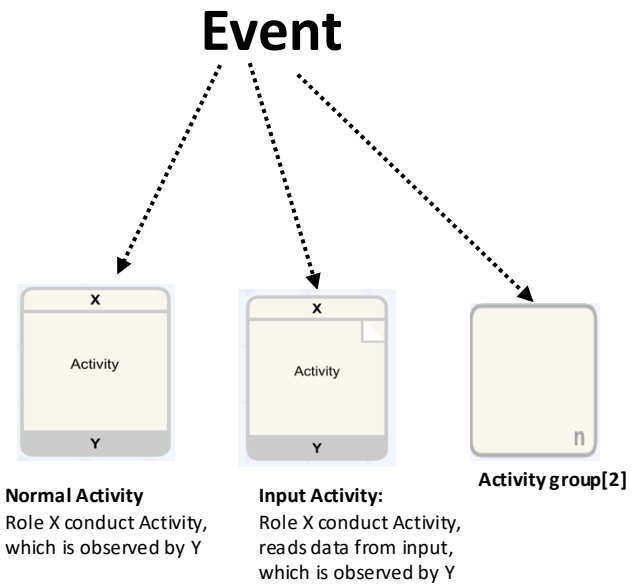
Overview Figure: Adapted from [1]

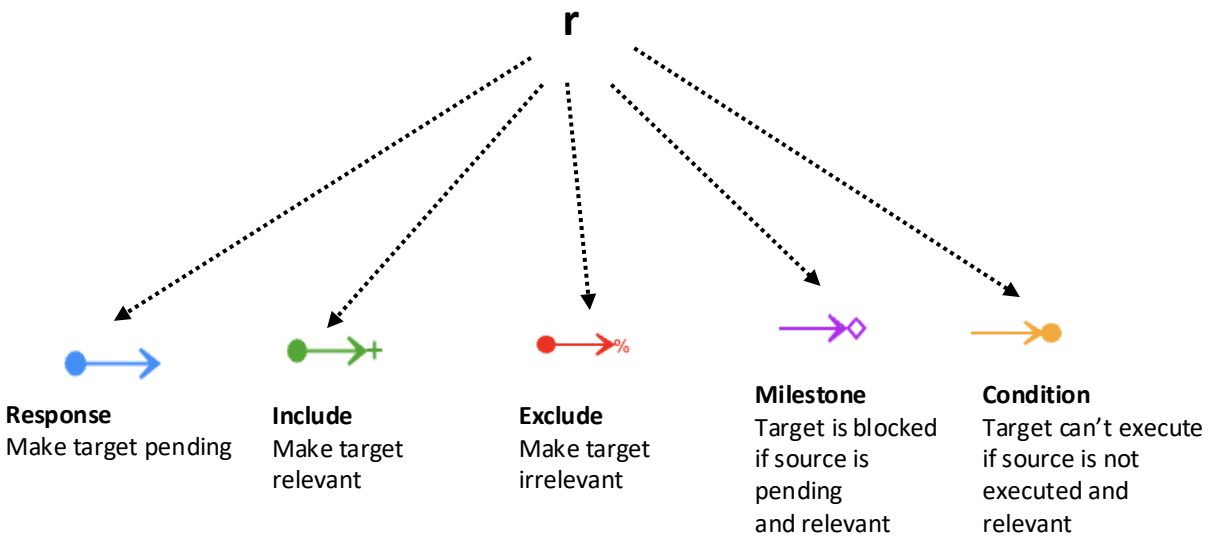
[1] Silva & Hildebrandt, "Human-Centric Hybrid-AI for No-Code Development," FSE 2025.

DCR Graph

DCR Graph: Dynamic Condition Response Graph[1].

$$(\mathbf{E}, \mathbf{R} \subseteq \mathbf{E} \times \{\rightarrow\!\diamond, \rightarrow\!\bullet, \bullet\!\rightarrow, \rightarrow\!+, \rightarrow\!\%\} \times \mathbf{E}, \mathcal{L} : \mathbf{E} \rightarrow \mathbf{A}, \mathbf{M} \in \mathcal{P}(\mathbf{E}) \times \mathcal{P}(\mathbf{E}) \times \mathcal{P}(\mathbf{E}))$$

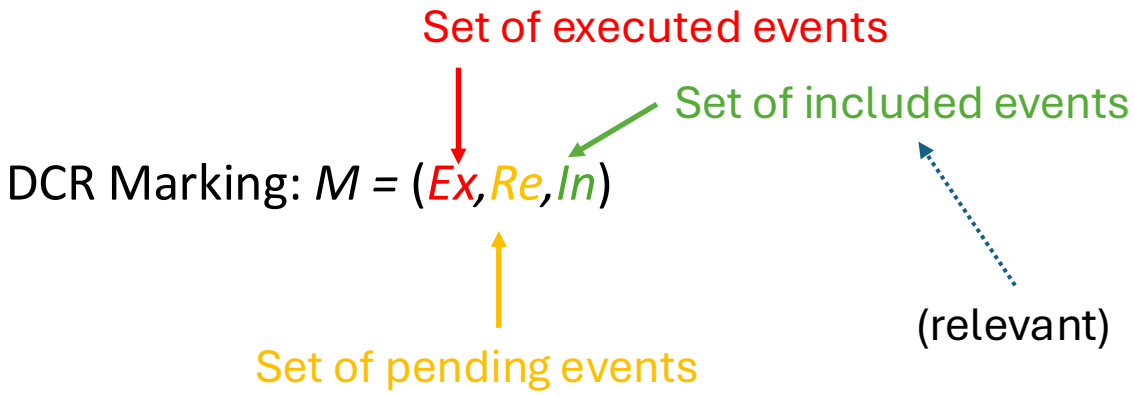




Note If an activity group is in a relation, all events in that group share the relation.

[1] Hildebrandt & Mukkamala, "Declarative Event-Based Workflow as Distributed Dynamic Condition Response Graphs," PLACES 2010, arXiv 2011.
[2] Hildebrandt et al., "Nested Dynamic Condition Response Graphs," FSEN 2011.

DCR Graph (execution semantics)



Event enableness :

Event e is enabled in marking $M = (Ex, Re, In)$ iff:

- **e is included**: $e \in In$
- **All e's conditions are executed or excluded**:
  $\forall e' \rightarrow e$, either $e' \in Ex$ or $e' \notin In$
- **No included pending is the milestone**:
  $\forall e' \rightarrow\!\!\diamond\, e$, either $e' \notin Re$ or $e' \notin In$

Transitions between DCR markings:

$M = (Ex,Re,In) \xrightarrow{e} M' = (Ex',Re',In')$

- e is enabled in M
- $Ex' = Ex \cup \{e\}$
- $Re' = Re \setminus \{e\} \cup \{e' \mid e \rightarrow e'\}$
- $In' = In \setminus \{e' \mid e \rightarrow\!\!\% \, e'\} \cup \{e' \mid e \rightarrow\!\!+ \, e'\}$

Accepting runs [1]:

$$M_0 \xrightarrow{e_0} M_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{n-1}} M_n$$

A finite $(e_0, e_1, \ldots, e_{n-1})$ is an accepting runs if:
In the end, there is no included pending events:
$Re_n \cap In_n = \emptyset$
represent regular languages

An infinite $(e_0, e_1, \ldots, e_{n-1})$ is an accepting runs if:
Any pending event must be evetually executed or excluded:
$\forall e \in Re_i, \exists j \geq i : e_j = e$ or $e \notin In_j$
Represent set of regular and omega-regular languages

[1] Mukkamala & Hildebrandt, "From Dynamic Condition Response Structures to Büchi Automata," TASE 2010.

DCR Graph Extensions

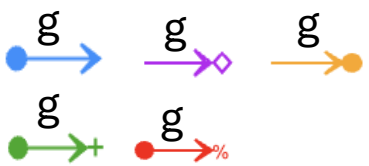
DCR graph with time constraint [1]

The target must be executed or excluded with deadline x

If source is included, before target is enabled, source must have been executed at least **x** time ago

Still regular language, one may verify everything with a big state space

DCR graph with data constraint (guard) [2,3]

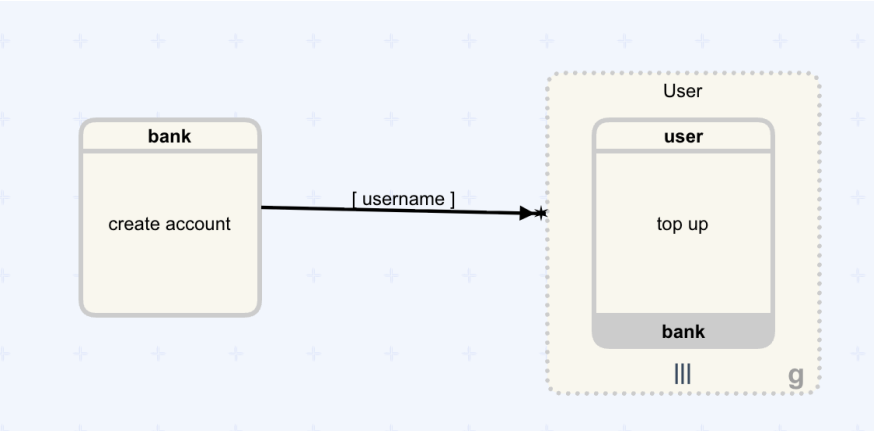


If g holds, relations take effect in the graph

Object centric DCR graph (dynamic sub processes) [4]

Allow activity group to created/deleted via object lifecycle

E.g. When creating account, a User activity group (identified by username) is created.

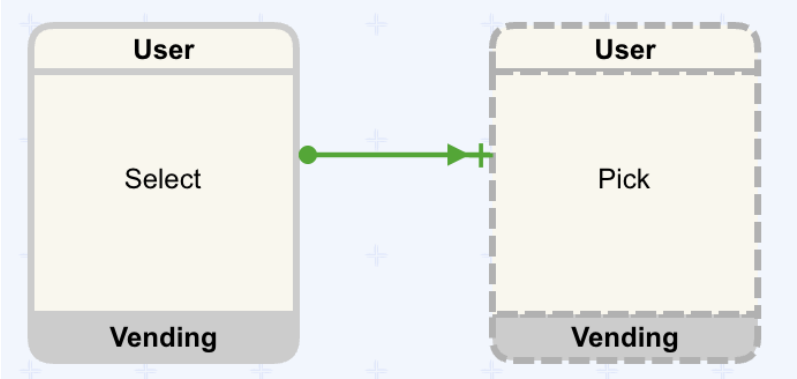


If unbound: OC DCR is turing complete [6], can't check for deadlocks.

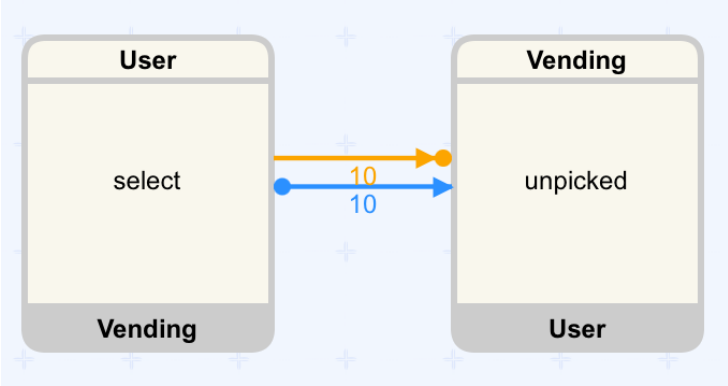
Static check can still be conducted via the liveness typing checking [5]

[1] Hildebrandt et al., "Contracts for Cross-Organizational Workflows as Timed Dynamic Condition Response Graphs," JLAP 2013.
[2] Hildebrandt et al., "Decision Modelling in Timed Dynamic Condition Response Graphs with Data," BPM Workshops 2021.
[3] Hildebrandt et al., "Declarative Choreographies with Time and Data," BPM 2023.
[4] Debois et al., "Safety, Liveness and Run-Time Refinement for Modular Process-Aware Information Systems with Dynamic Sub Processes," FM 2015.
[5] Debois et al., "Type-Checking Liveness for Collaborative Processes with Bounded and Unbounded Recursion," LMCS 2016.
[6] Debois et al., "Replication, Refinement & Reachability: Complexity in Dynamic Condition-Response Graphs," Acta Informatica 2018.

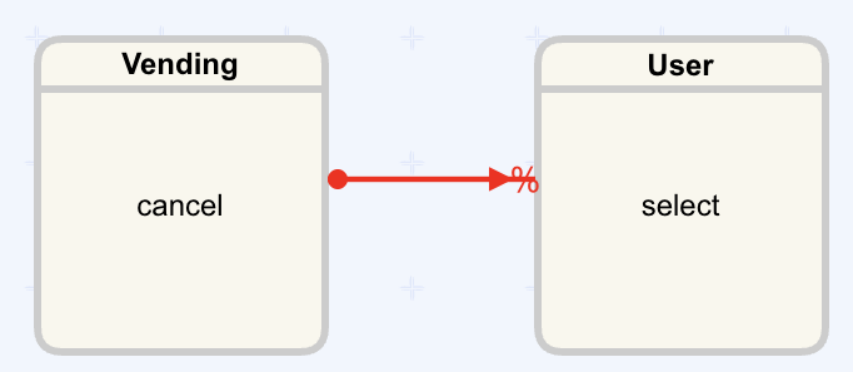
DCR Graph (Vending machine example with time)



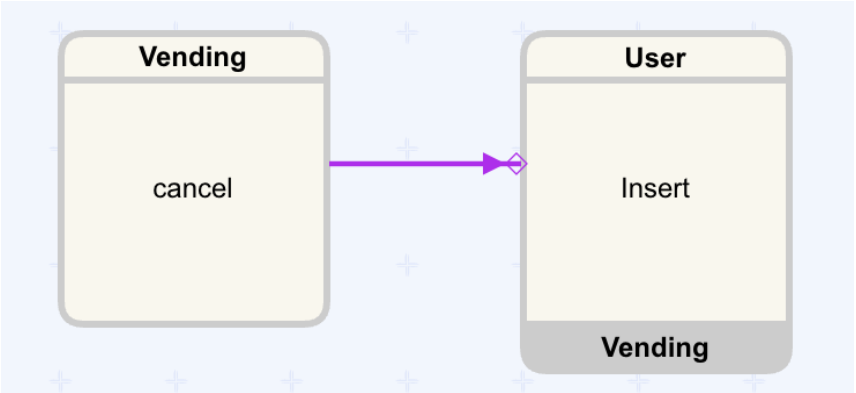
(a): Pick is initally set unavilable:Whenever user does **select**, **pick** become available



(b): 1. Whenever user does **select**, **unpicked** become pending with the deadline 10 time units
2. **unpicked** is not enabled if **select** is included and not executed at least 10 time units ago



(c): Whenever vending does **cancel**, select will not be accessible

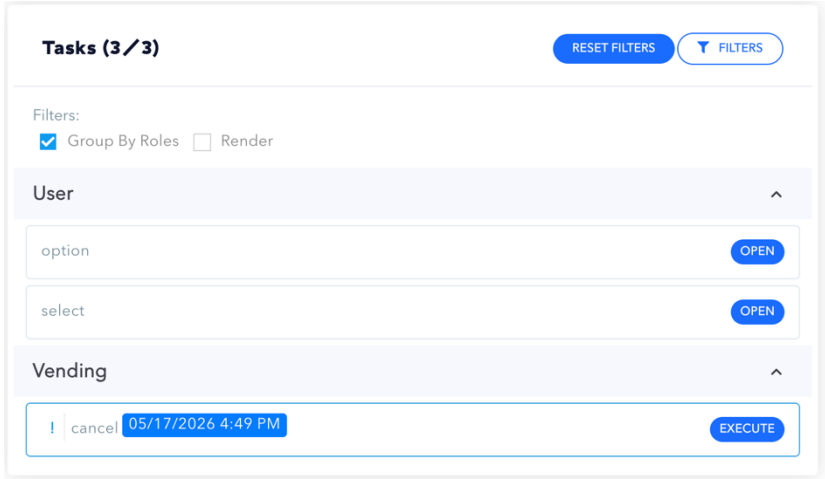


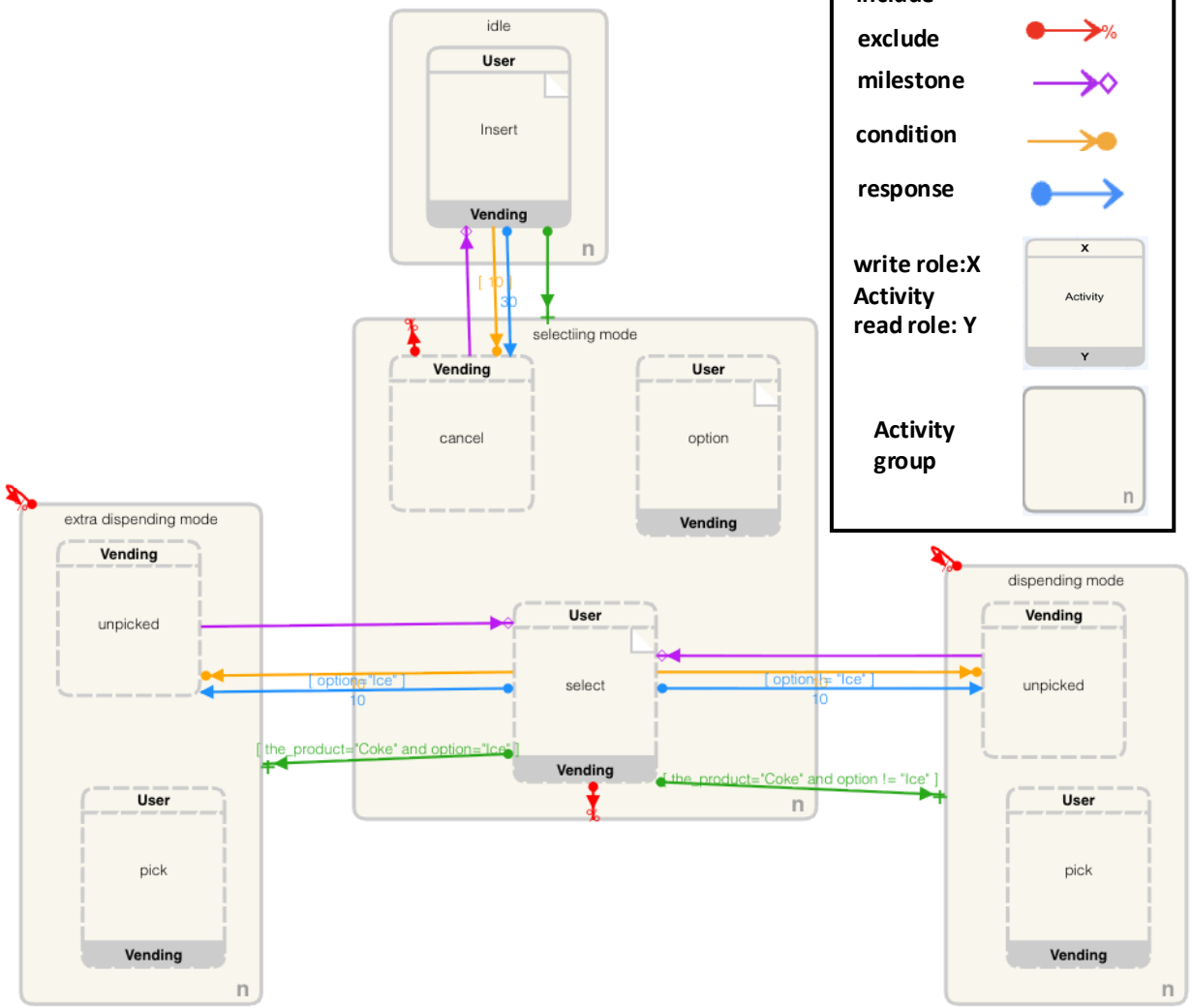
(d): While **cancel** is pending, **Insert** can't execute

DCR Graph(full vending machine example)

NL description:

- After the user inserts coins, the vending machine will automatically cancel if no operation is conducted within 30 time units
- If the user select coke with ice, the vending machine is in dispending mode
- If the user select coke with ice, the vending machine is in extra dispending mode
- After the user select, if the user doesn't pick up the product, the vending machine will unpick it.





https://www.dcrgraphs.net/tool/main/Graph?id=f35f2839-998c-4a81-b711-fcd51e3b3c51

Simulation tools to validate some event sequences (free academic account at DCRSolutions.net)

Works in progress (motivation)

DSL (Domain Specific Language) BDD (Behavior Driven Development)

BDD: used to specify the behavior of software

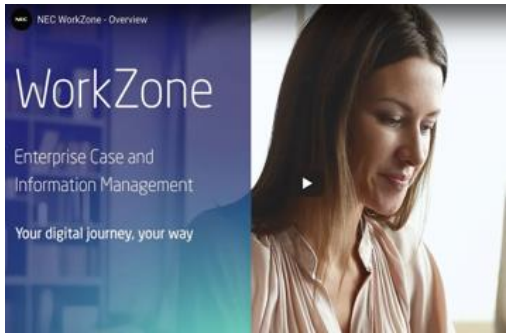
BDD scenarios (in Gherkin syntax):

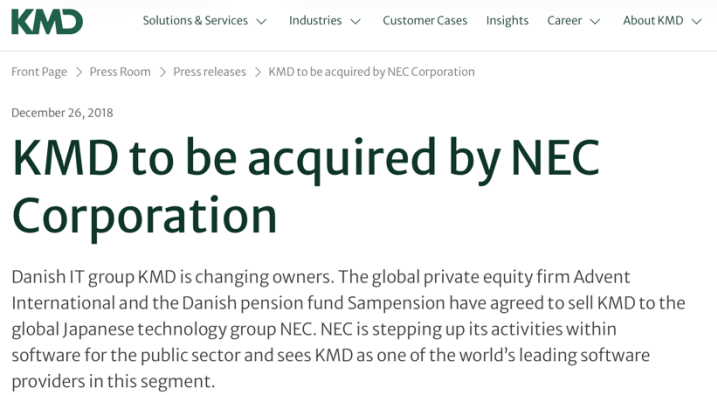
Scenario: **scenario name**
Given: Initial context
When : Activities that triger the scenario
Then: The expected outcome

Given the vending machine **is** idle
When I insert $2
Then the vending machine **is in** selection mode

- BDD is easy to understand across different roles: (business stakeholders, developer and domain experts)
- BDD is widely used in software engineers
- DCR has formal semantics (static and dynamic verification)
- DCR is widely used in Denmark (+70% of central government organisations use the KMD WorkZone EIM tool - with DCR embedded)

- Build a bridge between BDD and DCR graph




KMD Solutions & Services Industries Customer Cases Insights Career About KMD

Front Page > Press Room > Press releases > KMD to be acquired by NEC Corporation

December 26, 2018

KMD to be acquired by NEC Corporation

Danish IT group KMD is changing owners. The global private equity firm Advent International and the Danish pension fund Sampension have agreed to sell KMD to the global Japanese technology group NEC. NEC is stepping up its activities within software for the public sector and sees KMD as one of the world's leading software providers in this segment.

Works in progress

DSL (domain specific language) BDD (Behaviour-driven Development) grammar

model <domain **model** name>

entity <**entity** name> {
  initiator actions: <list of supported **initiator actions**>
  observed actions: <list of supported **observed actions**>
  data properties: <list of supported **data properties**>
  states: <list of supported **states**>
}

role <**role** name> {
  initiator actions: <list of supported **initiator actions**>
  observed actions: <list of supported **observed actions**>
}

declarative specification <**entity** name> is <**state** name> {
  IF [[<Entity State> /
    <Entity Property Assertion> /
    <Execution Description> /
    <Response Description>/
    <Inclusion Description>]
    (**AND**)]*
}

Scenario: <scenario name>
Given [[<Entity State> /
  <Entity Property Assertion> /
  <Response Description> /
  <Inclusion Description> /
  <Execution Description> / (**And**)]*

When <Entity Action>

Then [<Entity Property Assignment> /
  <Entity State>
  <Response Description> /
  <Inclusion Description> / (**And**)]*

Works in progress

DSL BDD → DCR BDD

Entity definition

```
entity vending machine {
   initiator actions:, cancel, unpicked
   observable actions: insert, select, pick
   states: idle, dispensing mode, selection mode,  extra
dispensing mode
   data properties: option, inserted amount, product }
```

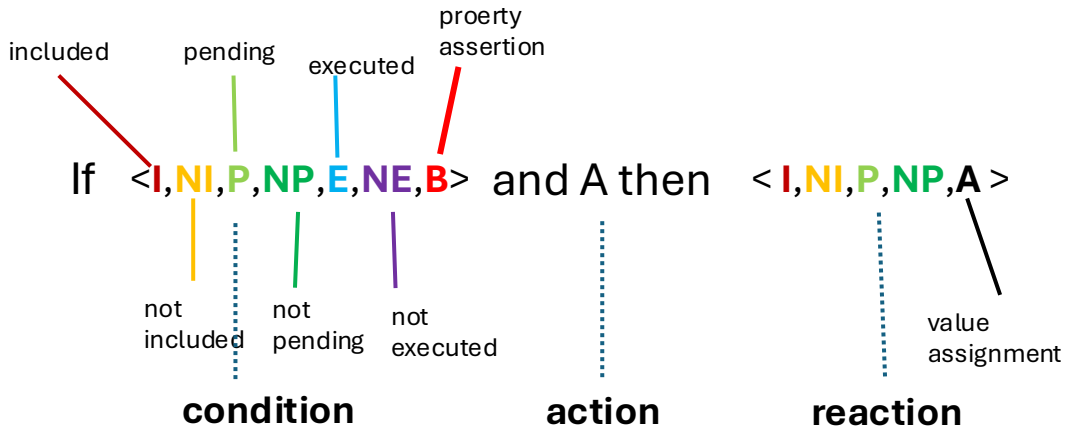
State Specification

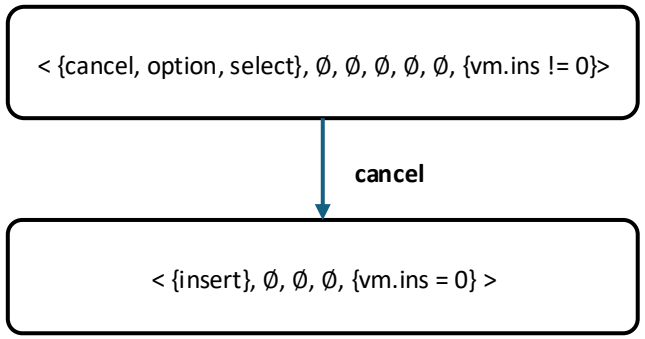
```
Declarative specification vending machine is idle mode {
        IF the action insert is available
        AND the inserted amount equal to 0
}

Declarative specification vending machine is selection mode {
  IF the action cancel is available
      AND the action option is available
      AND the action select is available
      AND the inserted amount does not equal to 0
}
```

BDD scenarios

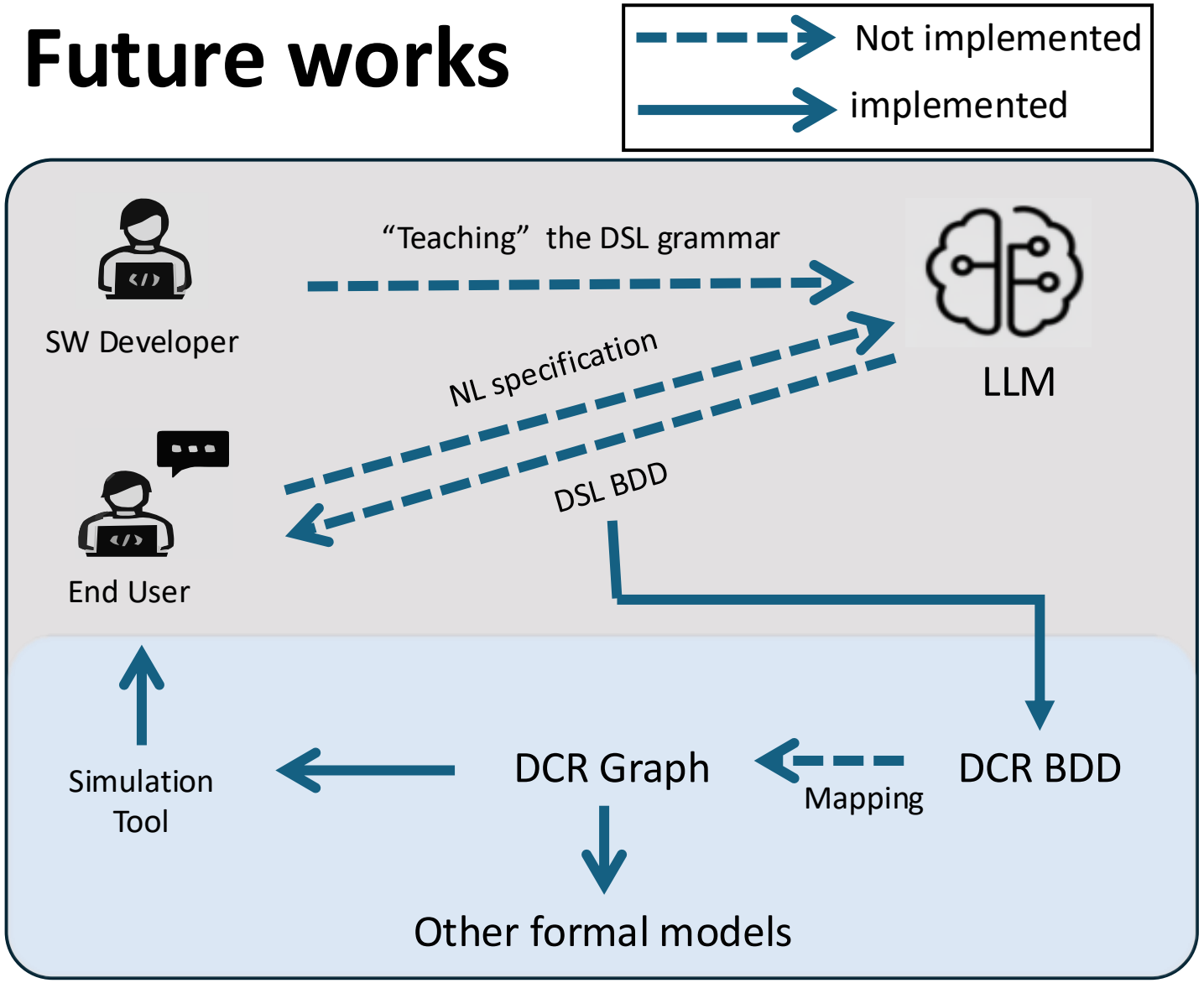
```
Scenario : Cancel
Given the vending machine is in selection mode
When the action cancel is executed
Then the vending machine is in idle mode
```

Generalised DCR rules (condition, action, reaction):





Future works



- Research on methodologies to generate DSL BDD via LLM (grammar constraint LLM)
- Further develop object centric DCR graph (to support ER diagram constraints)
- Allow DSL BDD to support object centric DCR graph features
- Research on DCR BDD to DCR graph mapping

Thanks for your attention!

Contact: Xinyuan Tu (xinyuan.tu@di.ku.dk), Thomas Hildebrandt (hilde@di.ku.dk), Thiago Rocha Silva (trsi@mmmi.sdu.dk)