Overview of our research in software modeling and simulation

Prof. Eugene Syriani
Overview of our research in software modeling and simulation – Eugene Syriani, 2022
Outline
Domain-specific IDEs – Editors

Graphical syntax-directed editing

Textual parser-based editing

Projectional editing
Domain-specific IDEs – Interaction

Overview of our research in software modeling and simulation – Eugene Syriani, 2022
Domain-specific IDEs – Storage

- Storing models in graph databases
- NeoModelingFramework
Collaborative modeling – Multi-view consistency

Model-data consistency

Tolerating inconsistency

Multi-view modeling
Collaborative modeling – Real-time collaboration

- CollabServer
- CRDT
- lowkey
Collaborative modeling – Model versioning

\[ \Delta_M(G_1, G_2) = \sum_{m_1 \in Mov_1, m_2 \in Mov_2} \delta_M(m_1, p(m_2)), \text{ where } l(m_1) = l(m_2) \]

Element distance

\[ \Delta_E(G_1, G_2) = \frac{\{|v_1 \in V_1 | \exists v_2 \in V_2, l(v_2) = l(v_1)| + \{|v_2 \in V_2 | \exists v_1 \in V_1, l(v_1) = l(v_2)|}{|V_1| + |V_2|} \]

Value distance

\[ \delta_V(v, x) = \begin{cases} |a(v_1, x)| & \text{if } a(v, x) = 0 \\ |a(v, x) - a(v_1, x)|/a(v, x) & \text{otherwise} \end{cases} \]
Overview of our research in software modeling and simulation – Eugene Syriani, 2022

Model transformation – Domain-specific transformation languages

Custom transformation languages

Higher-order transformations

T-Core

MoTif

McGill

T-Core

MoTif

User

Controller

Automatic

User Controlled

graph_in
ccontrol_out
g_init
done_rcv
step_out
g_init
done_send
step_in
g_unmodified
control_in
notify
key_out

graph_in

c

g_init
done
step

g_modified

key_in

Controller

User

Automatic

User Controlled

Higher-order transformations

MoTif

McGill

T-Core

Custom transformation languages

Higher-order transformations

Overview of our research in software modeling and simulation – Eugene Syriani, 2022
Model transformation – Design patterns

Detection of variants

Refactoring transformations by-example

Exception handling

[t10, Trace, f]
[t11, source, c]
[t1, ExprStmt, p]
[t2, dm, _]
[t3, Trace, c]
[t4, target, c]
[t5, Transition, c]
[t6, dm, _]
[t3, Trace, c]
[t7, source, c]
[t1, ExprStmt, p]
[t8, expression, c]
[t9, IdentifierRef, c]
[t12, dm, _]
[t10, Trace, f]
Model transformation – Code generation

Template-based code generation

Design-time input

Class
name: string

uses

conforms to

Runtime input

Person

Template

<% content class%>
public class <%=name%> { String id; }

Template engine

input

generates

Output

public class Person { String id; }

GASTM Metamodel

RPG Metamodel

manual refinement

conforms to

codes generation

DSL

Atlo

DSL

codes generation

workflow section with our collaboration

Modernization of legacy software

Overview of our research in software modeling and simulation – Eugene Syriani, 2022
Outline

- Digital twins
- Adaptive systems
- Discrete event Simulation
- Model recommenders
- Models and data
- Data curation
  - System learns from its environment
  - Prescriptive model to enact decisions on system
- Learning simulation models
- Integrating IoT data to infer model
- Modeling of bio-CPS

Overview of our research in software modeling and simulation - Eugene Syriani, 2022
Adaptive systems – Simulation

Distributed DEVS simulators

Traffic simulation

Robotics
Overview of our research in software modeling and simulation – Eugene Syriani, 2022

Meta-model recommenders

Models and data reference framework

Intelligent modeling assistance
Outline
Systematic reviews
Upcoming projects

Bio-Cyber-Physical System modeling and simulation

*Vertical farming*

Integration of heterogeneous data for AI model validation

*Water toxin prediction*

Learning DEVS simulation models

*Digital twins*
Overview of our research in software modeling & simulation

- Eugene Syriani, 2022