



An Online Modeling Language For The Low-Code Development Of BDI Agents : LCD4BDI

Burak Çelik

Outline

- What is Simplified Agent Modeling Language(SAML)?
 - Overview
 - Syntaxes
- What is Low-Code Development For BDI(LCDP4BDI)?
 - Overview
 - Why Do We Need?
 - Syntaxes
 - Demonstration

What is Simplified Agent Modeling Language(SAML)?

Overview

Purpose of SAML

- Developed as a design language framework.(M2M-M2C and model level debugging)
- Serves to model essential elements for MAS (Multi-Agent System) design.

Components of SAML Meta-Model

- Meta-model includes basic elements required for MAS design.
- Elements for modeling agent-environment interactions are present.

Key Entities in Meta-Model

- MAS meta-entity represents the system to be modeled.
- Each MAS must have at least one Agent and Environment meta-entity.
- Ability meta-entities can represent capabilities of agents within the system.
- Abilities may interact with one or more environments.
- Abilities may possess sub-abilities and meta-entities such as Plans and Beliefs.











What is Simplified Agent Modeling Language(SAML)?

Syntaxes

- The concrete syntax of the language consists of four different diagrams representing four different perspectives:
- MAS, Environment, Ability, and Plan diagrams.
- The MAS diagram, which depicts the overall structure of the system, is responsible for modeling agents, their capabilities, environments, and their interactions.
- Environment, Ability, and Plan diagrams are responsible for modeling the internal structure of the environment, capabilities, and plans respectively.

What is Simplified Agent Modeling Language(SAML)?

Syntaxes

Kavram	Gösterim	Kavram	Gösterim
Eylem		Olay	
Etmen		MAS	
İnanç		Mesaj	
Yetenek		Operasyon	
Ortam		Plan	

What is Low-Code Development For BDI(LCDP4BDI)?

Overview- Why Do We need?

- Originally based on SAML, DSML4BDI but lighter.
- However, the utilization of its modeling perspectives requires infrastructure dependent on Eclipse desktop components and libraries.
- With LCDP4BDI, the language will be expanded, enriching the modeling environment, and the usage of the language syntax will become entirely online. Thus, with low-code development techniques, modeling of BDI agents will be possible in a platform-independent manner.
- On the other hand, offering the DSML4BDI and SAML language as a Software as a Service (SaaS) model online could contribute to experiencing the use of low-code development techniques in developing autonomous systems.

What is Low-Code Development For BDI(LCDP4BDI)?

Overview- Why Do We need?

- Originally based on SAML, DSML4BDI but lighter.
- The LCDP4BDI aims to extend the DSML4BDI/SAML language and enrich the modeling environment to enable the modeling of BDI agents in a platform-independent manner using low-code development techniques.
- Additionally, emphasizing the significance of the DSML4BDI/SAML language in the software modeling domain, it is noted that this language could contribute to the software modeling field as a web-based service.
- The platform to be developed in the LCDP4BDI will undergo an examination of web-based software development languages, integration of third-party packages, and data storage technologies to provide multi-platform support.
- Similar to the desktop version of the DSML4BDI/SAML language, the online platform will facilitate developers in creating MAS models through an online palette.
- With LCDP4BDI, the language will be expanded, enriching the modeling environment, and the usage of the language syntax will become entirely online. Thus, with low-code development techniques, modeling of BDI agents will be possible in a platform-independent manner.

What is Low-Code Development For BDI(LCDP4BDI)? Syntaxes

The screenshot displays a Low-Code Development Environment (LCDP4BDI) interface for a project named "Web Flow DSML". The main workspace shows a diagram titled "CLEANINGPROCESSSTART.PLN" with a grid background. The diagram consists of several nodes and edges:

- Relation Nodes (Grey):** GW:gotoGarbage, Cleaning:picked(grb), GW:goToBurner, Cleaning:dropped(grb), and GW:startBurn.
- Action Nodes (Orange):** goToGarbage, pickGarbage, goToBurner, and dropTheGarbage.
- Message Node (Blue):** startBurn.

Connections between nodes are labeled with relationship types:

- GW:gotoGarbage (relation) is connected to goToGarbage (action) via a "Use" edge.
- Cleaning:picked(grb) (relation) is connected to pickGarbage (action) via an "AddBelief" edge.
- GW:goToBurner (relation) is connected to goToBurner (action) via a "Use" edge.
- Cleaning:dropped(grb) (relation) is connected to dropTheGarbage (action) via a "DelBelief" edge.
- Cleaning:dropped(grb) (relation) is connected to startBurn (message) via an "AddBelief" edge.
- GW:startBurn (relation) is connected to startBurn (message) via a "Use" edge.

The interface includes a "Project Files" sidebar on the left, a "Designer Tool Box" on the right with "Nodes" and "Edges" sections, and a "save" button in the top right corner of the workspace.

What is Low-Code Development For BDI(LCDP4BDI)? Demonstration

- Garbage Collector
- <https://web-dsml.vercel.app/>