
# Eclipse Sketch

## Creation Review

Planned Review Date: April 28, 2010
Communcation Channel: eclipse.sketch forum

Ugo Braga Sangiorgi
Mariot Chauvin

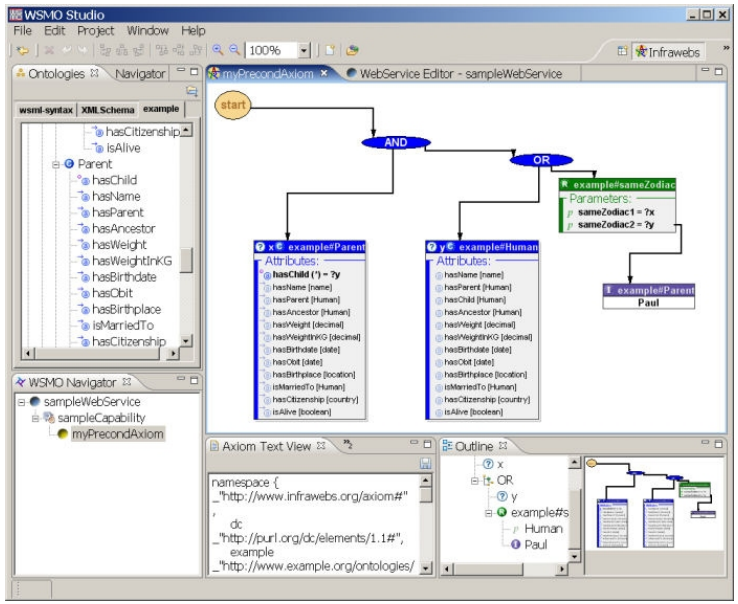
# Background

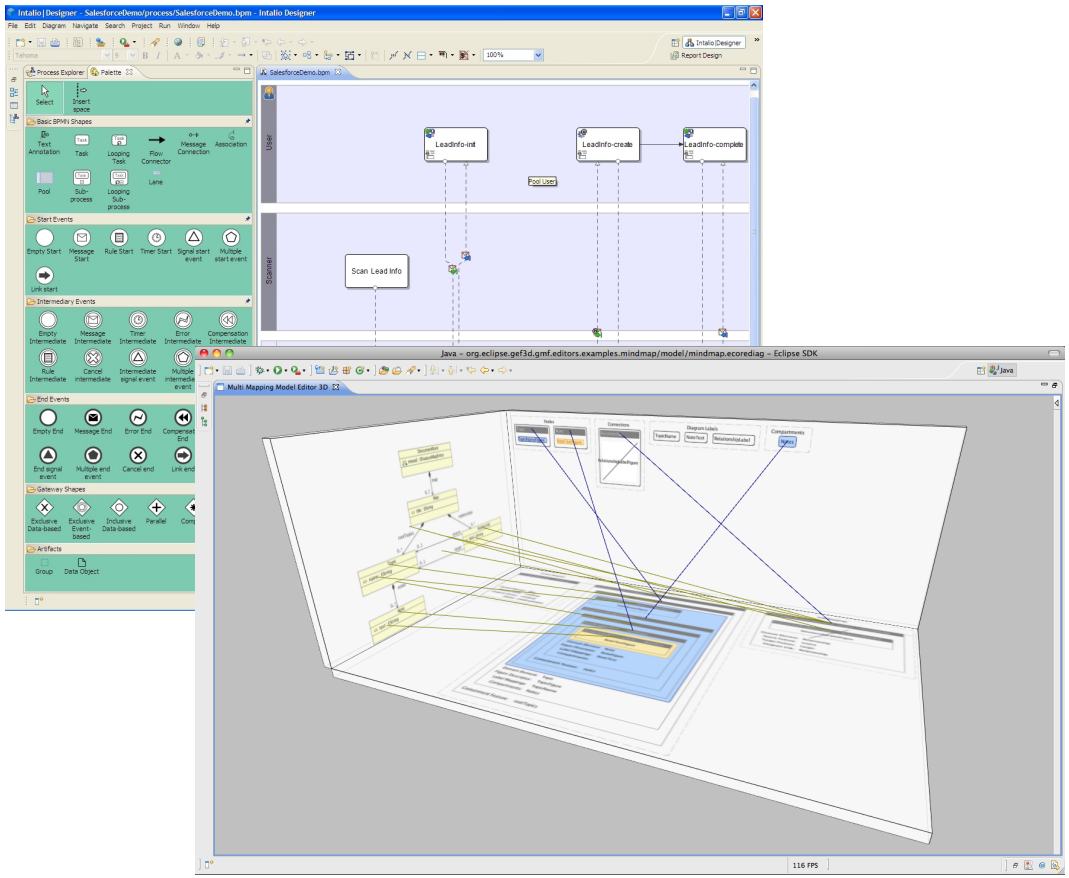
Today, plenty of devices supporting touch and pen are available, providing new ways of interaction.

However, little support is made available to take advantage of those new ways and applications are used with the classic point-and-click paradigm

# Background

Modeling is an activity that might benefit from pen interaction as well, and there are many modeling editors built with Eclipse frameworks



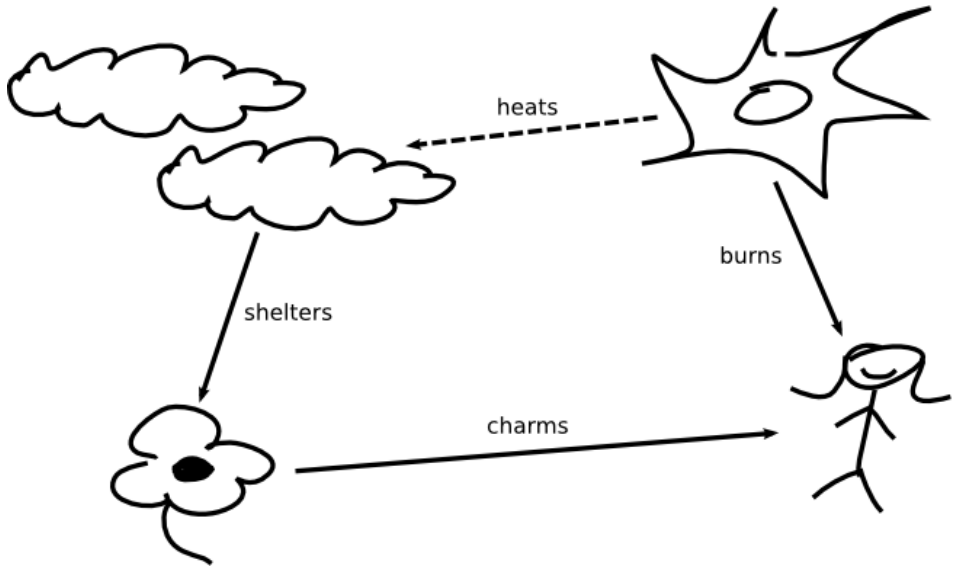

 

# Project Scope

## What is Sketch?

A tool to **enrich** user's interaction with Eclipse graphical editors, supporting a more **flexible** modeling activity by taking advantage from the capabilities provided by touch/tablet devices using free-hand sketching recognition.

---




"Flower" might have infinite graphical representations:

Approach described here

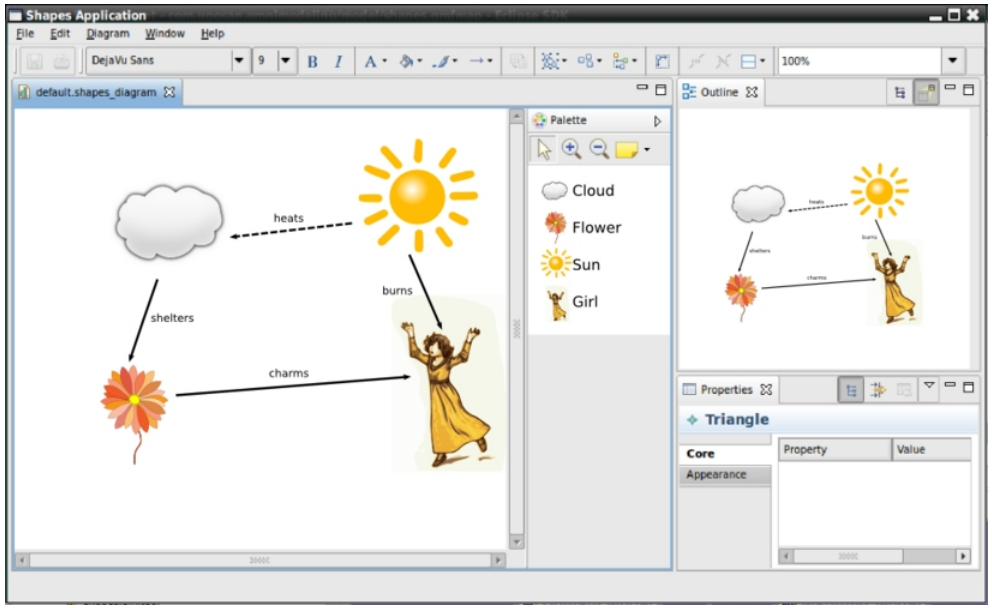
 

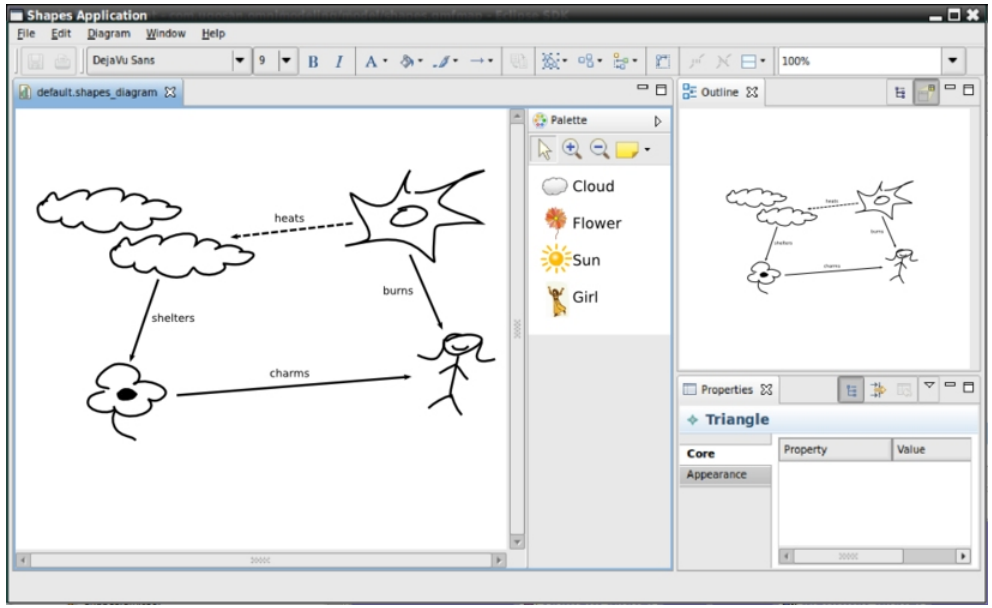
# Project Scope

**Goal 1**

To allow the creation of graphical editors flexible enough to hold **any** visual representation, still constructing a model underneath.

“beautified” vs. “sketched”

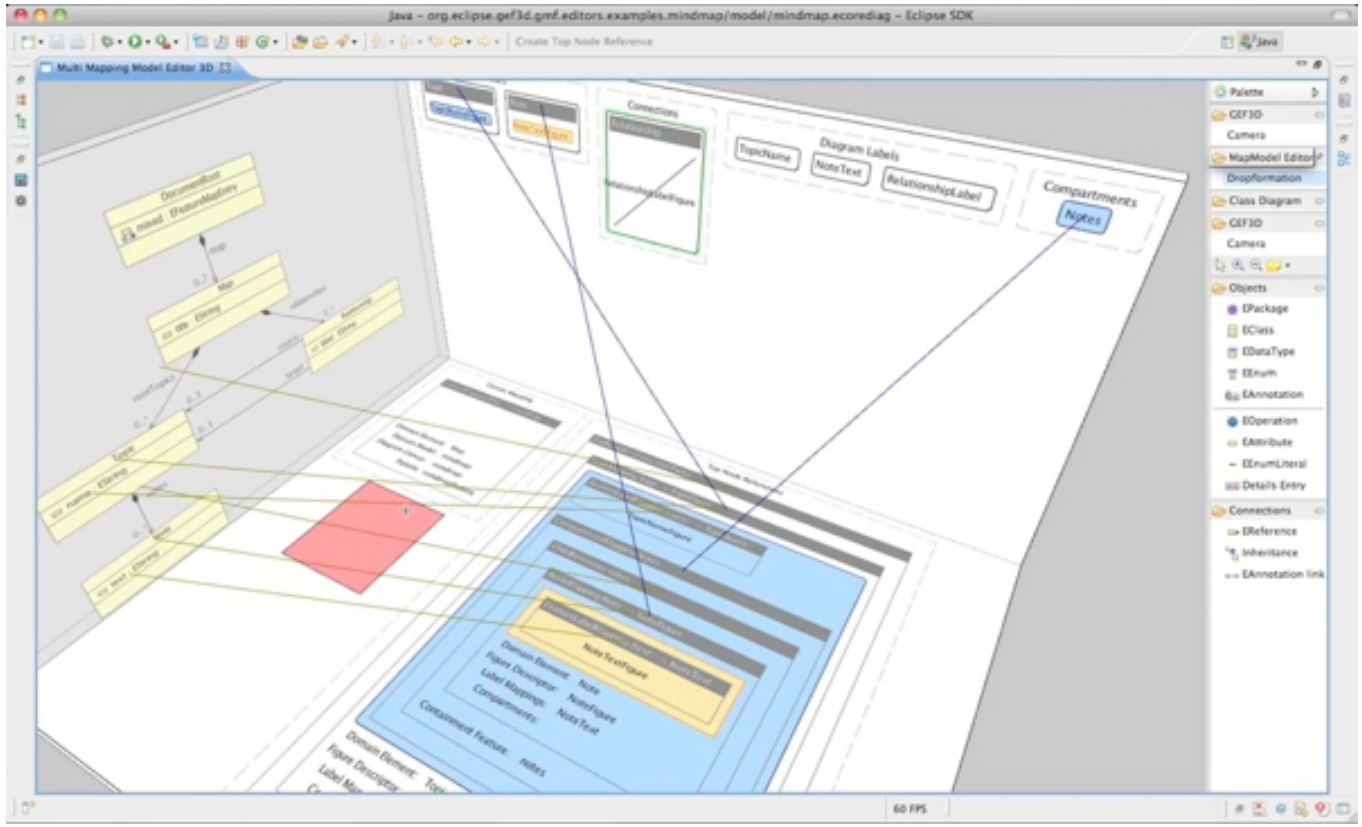


# Project Scope

**Goal 2**

To provide **gesture** recognition, by transforming gestures into commands such as move, rotate, resize, copy to GEF and GMF editors.



# Driving Values

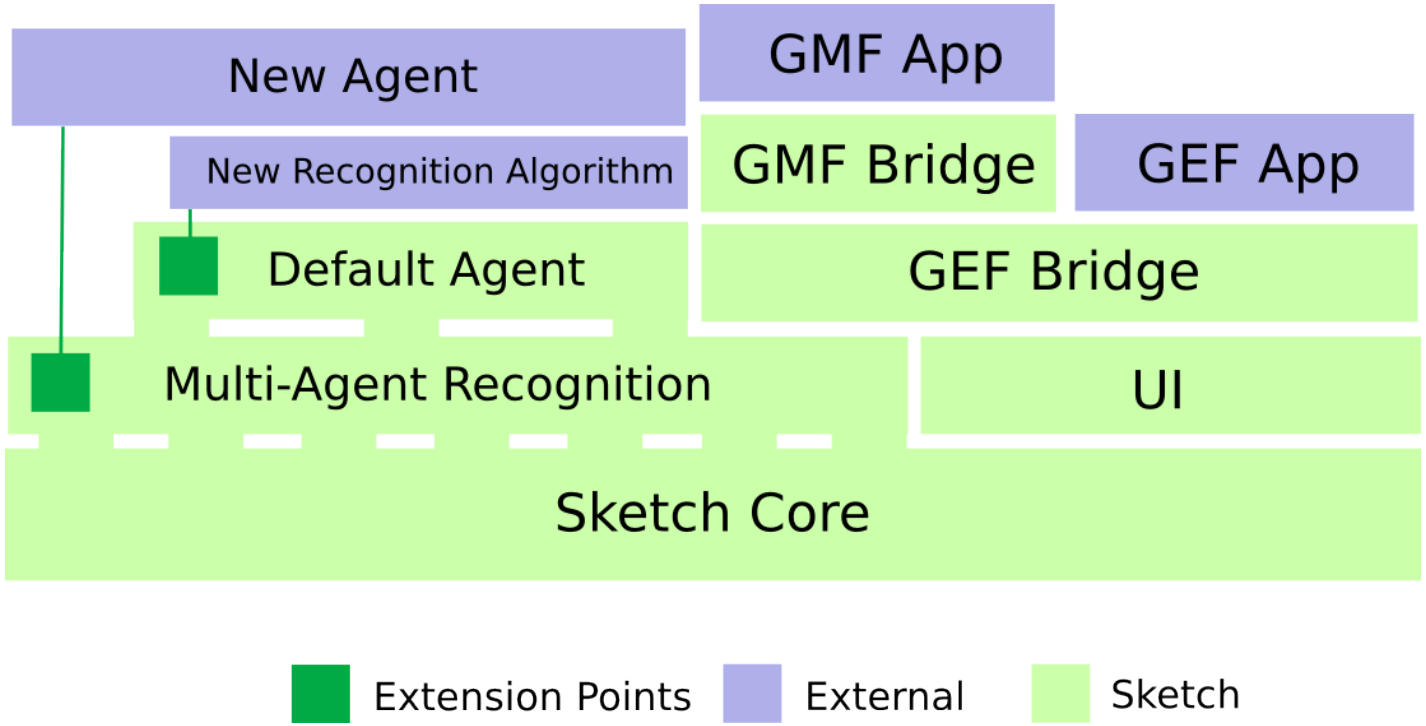
**Simplicity** Developers must be able to add sketching capabilities to their existing GEF/GMF editors by using extension points and little code

**Flexibility** The API should let developers write their own recognition algorithms, using extension points. Also should let developers use just a part of the API (gestures or shape recognition, for instance)

**Stability** The API is intended to learn from user's input. Thus it must inform users about the current state of the recognition mechanism, letting them to perceive flaws and eventually roll back to a correct state

# Proposed Components




- **Core** – holds the multi-agent recognition engine, must be extensible to hold new agents and new algorithms
- **GEF bridge** – uses AbstractTool
- **GMF bridge** – uses AbstractTool and recognizes IElement's
- **UI** – provides views to configure recognition parameters, gesture commands and visualize sketches

# Relationship to other Eclipse Projects

Sketch is initially built with dependencies from both GMF and GEF. Will progressively evolve to separate the dependencies (“bridges”).

Will directly benefit:

* GEF3D - supporting gestures
* GEF - allowing sketching + annotations
* GMF - allowing sketching + annotations + flexible visual model

# Community Feedback


**Interested parties**

Obeo (http://www.obeo.fr)
Aurélien Pupier, Mickael Istria (BonitaSoft)
Oisín Hurley
David Sciamma (Sierra Wireless)
Jens von Pilgrim, Kristian Duske (GEF3D)
Carsten Reckord, Andreas Scharf (Yatta Solutions)
Simone D.J. Barbosa (PUC-Rio)
Eraldo R. Fernandes (PUC-Rio)
Chris Aniszczyk
Ed Merks
Miles Parker (AMP, Metascape)

**FlexiTools'10 workshop paper** (link)

# Initial Committers

## Ugo Sangiorgi (independent)

**Bio** Ugo's professional experience includes software development and consultancy on Eclipse-based products and plug-ins, both Open Source and commercial. He has a M.Sc degree in Human-Computer Interaction by the Informatics Department of the Pontifical Catholic University of Rio de Janeiro (PUC-Rio). He is the creator of the Sketch API for Eclipse, developed during his research with flexible modeling tools and interaction design.

## Mariot Chauvin (Obeo)

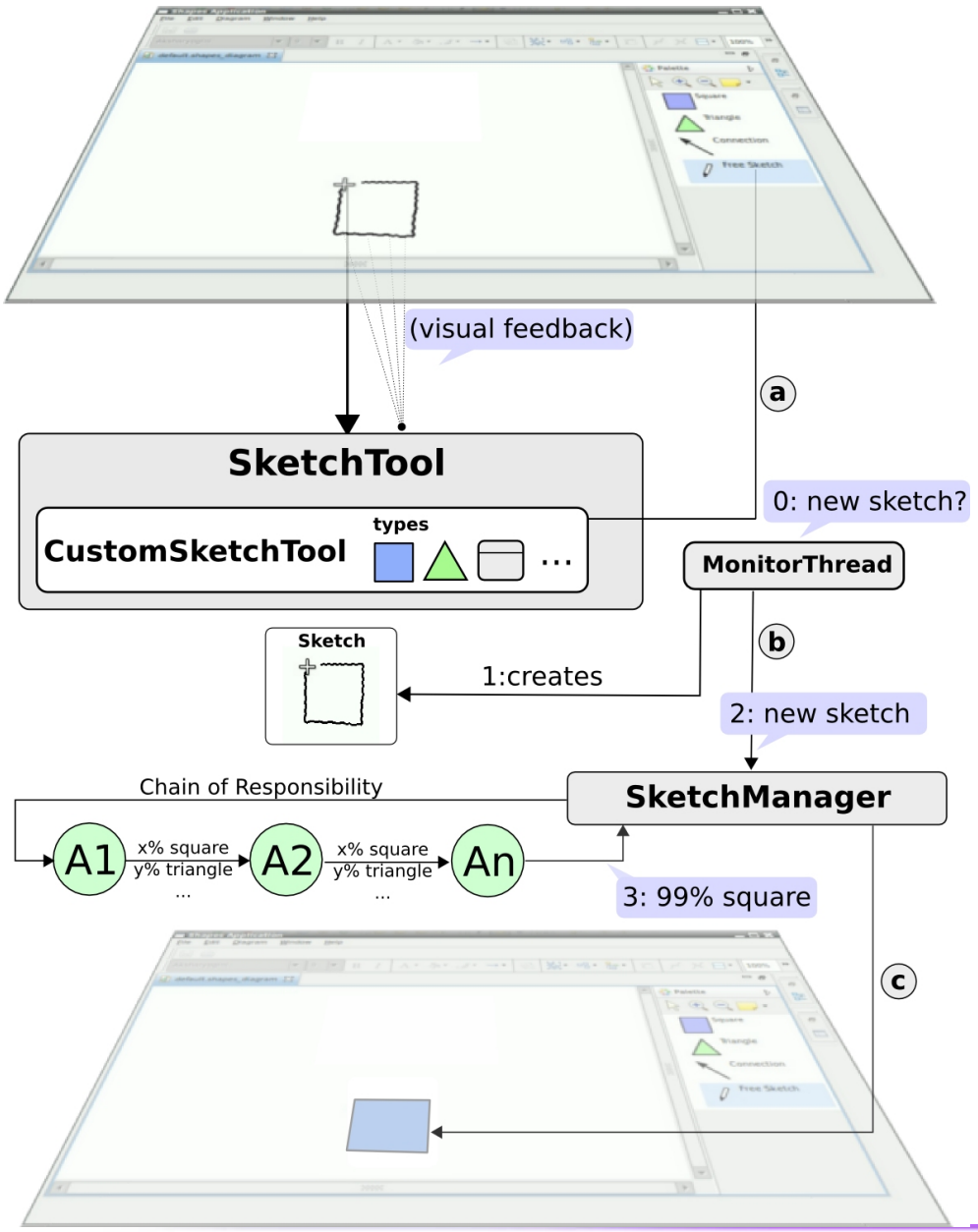
**Bio** Mariot is a Software Engineer working for Obeo on Model Driven Engineering. He is one of the main contributor to Obeo Designer product. He is a GMF Committer and has recently become SWTBot committer. During 2007 summer he participated to the Google summer of code for the Eclipse project, he worked on seamless debugging for Java-Jni applications, and presented his work at EclipseCon 2008

# Mentors

* Chris Aniszczyk
* Ed Merks

# Code Contribution




**First contribution**

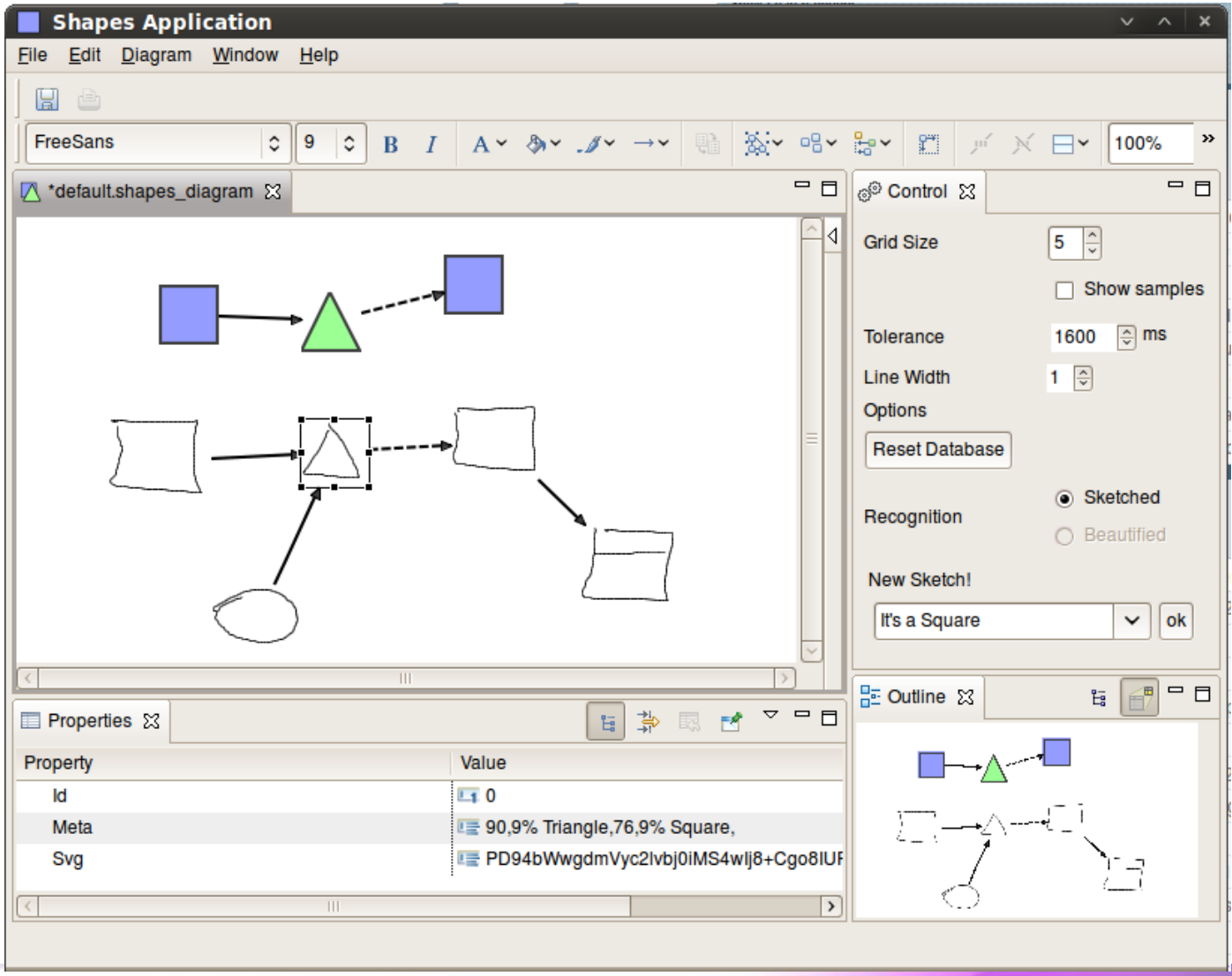
- Engine to detect and create elements defined in GMF editors

- Sample GMF editor (Shapes Application) with basic shape recognition and creation

- Initial UI for recognizer's configuration

# Code Contribution

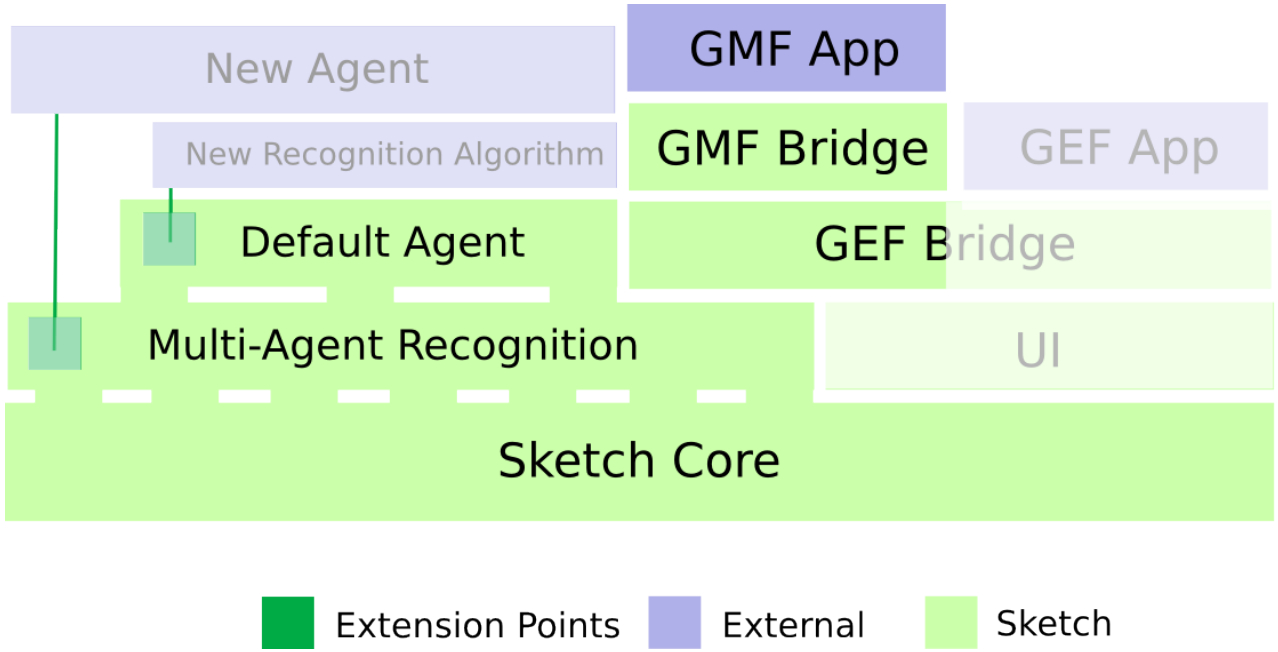
- Shapes Application shows elements with flexible graphical representation and it is also able to learn new shapes



# Code Contribution

## Current state




## Immediate next steps

- Create and structure the extension points
- Separate GMF and GEF bridges
- Create UI plugin
- Call community to contribute with new algorithms

# Plan

- First release by the end of June
- Not yet part of 3.6. Might be part of 3.7 release train
- EclipseCon 2011 presentation

# IP Concerns

- Code is not published anywhere yet
- Code license will be EPL
- No third-party dependencies