Java Workflow Tooling (JWT)
Release review: JWT v0.7

Marc Dutoo (Open Wide, FR)
Florian Lautenbacher (University of Augsburg, DE)
Christian Saad (University of Augsburg, DE)

Overview

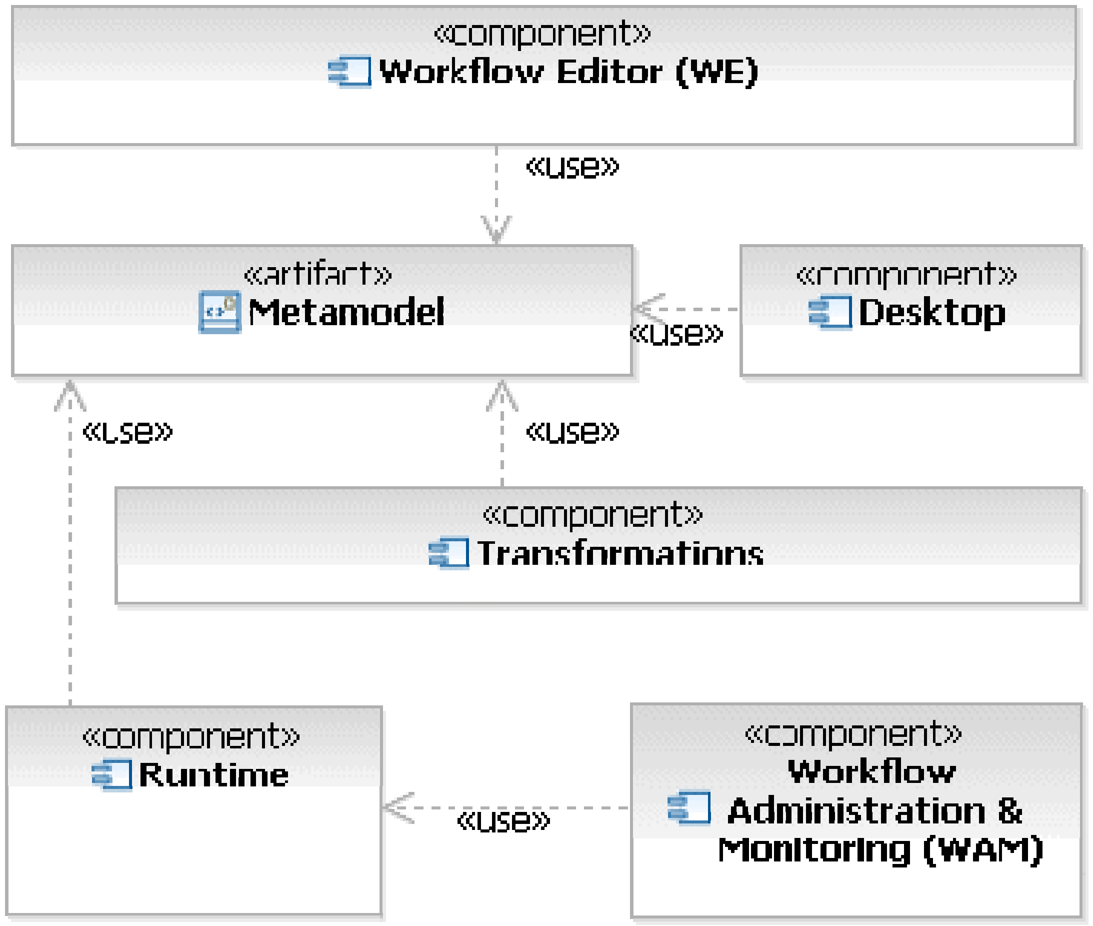
Introduction

- Java Workflow Tooling (JWT) is a technology sub-project currently in Incubation phase.
- JWT aims at providing a complete Business Process Management (BPM) and workflow tooling platform
  - with a special focus on a unified approach to BPM design, allowing to bridge the gap between BP representation, BP standards, BP engines, BP deployment environments (platforms, Information System, SOA)
- This release (version 0.7) covers improvements, fixes and additional features in the Workflow Editor and a separate meta model plugin which is now independent from the Workflow Editor.
- These slides conform to the Eclipse Guidelines for a Release review which can be found under http://www.eclipse.org/projects/dev_process/release-review.php.

About JWT - Goals of JWT

- Support for workflow and business process modeling, deployment, execution and monitoring inside Eclipse
- Provide a complete, flexible, interoperable and usable BPM toolkit
- Set of generic and extensible plugins and APIs
- Extensions allowing support for specific business representations, process language formats, process engines, service platforms, etc.
- Targeting and supporting SOA in close collaboration with the Eclipse STP project

About JWT - Components of JWT *



* This overview is intended to show the architecture of JWT and does not speak about components in the Eclipse terminology

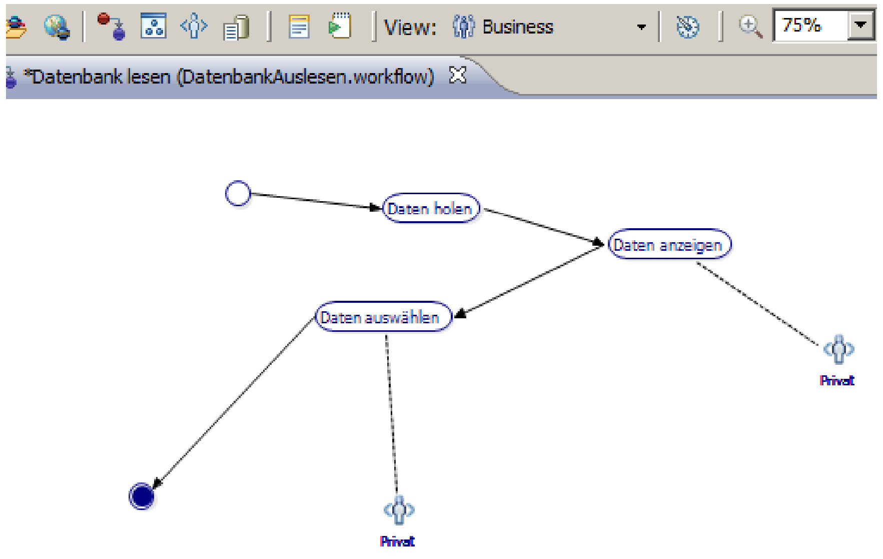
Features in this release - Overview

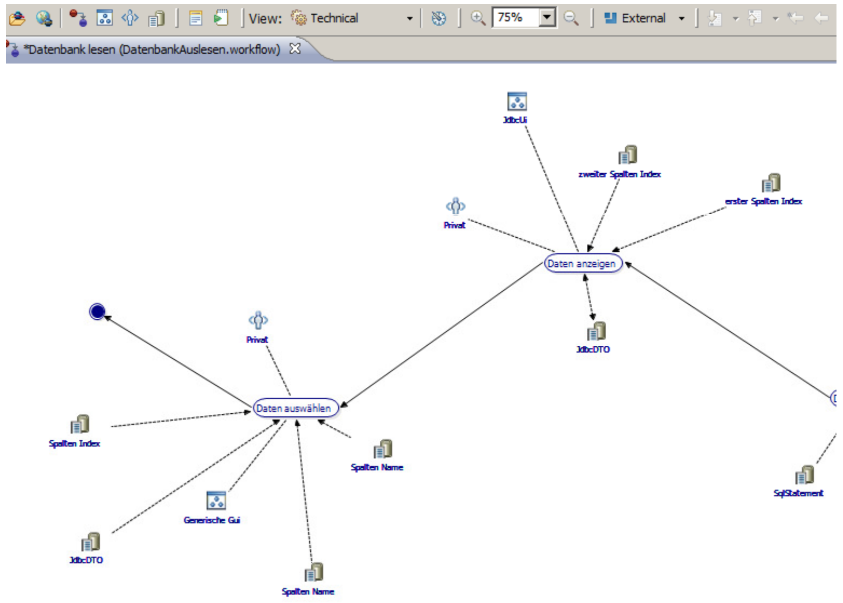
- **Workflow Editor (WE)**
  - A new view meta model that extends the main model (see below)
  - View information is stored in a separate file
  - Location/Size information of graphical elements can be set independently for each view
  - Included automatic layout algorithms from the GEF Zest project
  - New ATL based converter for old workflow files
  - Many big&small bugfixes

- **Meta Model**
  - Meta model is now independent from the Workflow Editor
  - View information (location, size, helper classes for displaying certain graphical elements) has been removed from this core model
  - It consists of the model itself as well as edit code to allow the manipulation of workflows outside the Workflow Editor
  - Resources like icons can be used through a dependency injection mechanism

Features – Workflow Editor

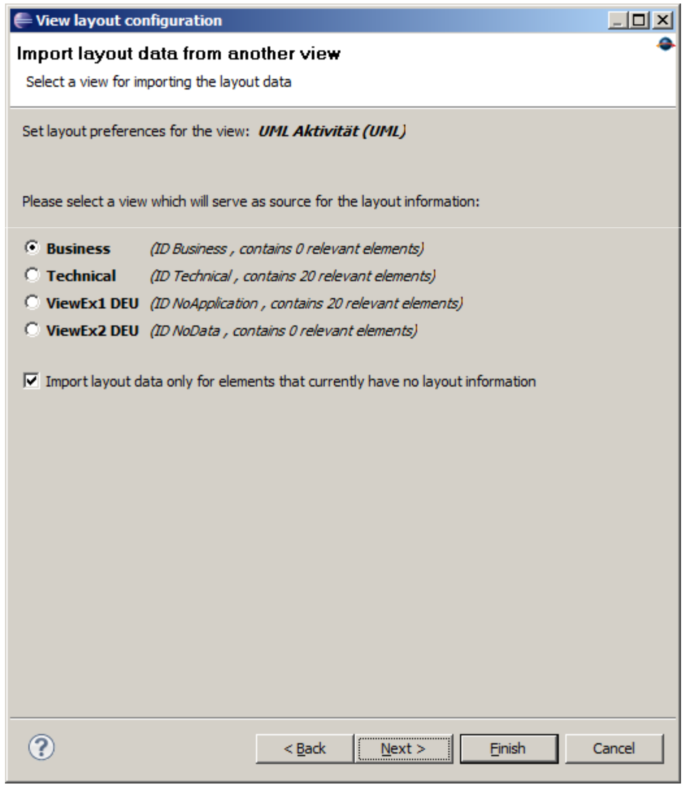
Objects can now have different locations/sizes in different views





Features – Workflow Editor

Layout data can be imported from other views or generated using GEF Zest layouting algorithms


View layout configuration

Import layout data from another view

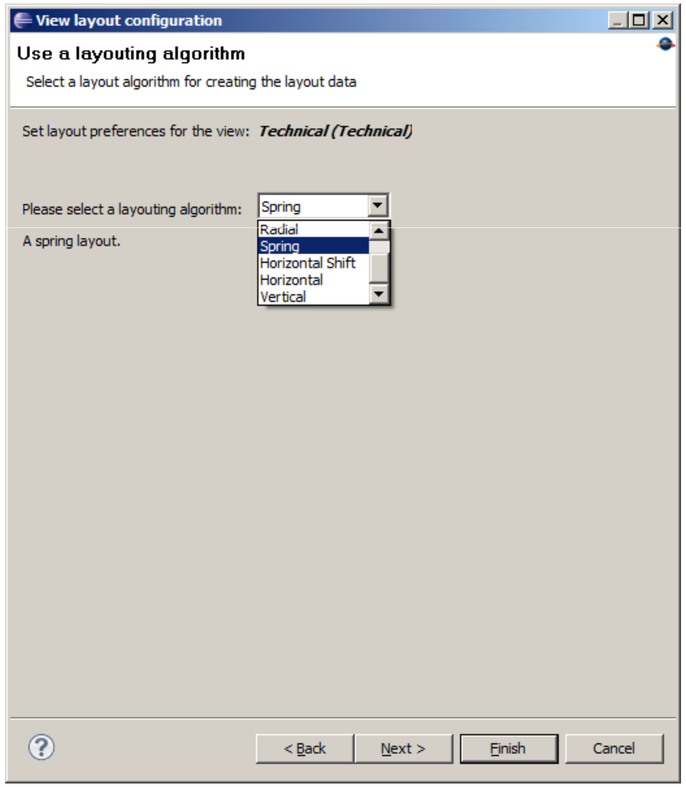
Select a view for importing the layout data

Set layout preferences for the view: ***UML Aktivität (UML)***

Please select a view which will serve as source for the layout information:

- [✓ **Business**] *(ID Business , contains 0 relevant elements)*
- **Technical** *(ID Technical , contains 20 relevant elements)*
- **ViewEx1 DEU** *(ID NoApplication , contains 20 relevant elements)*
- **ViewEx2 DEU** *(ID NoData , contains 0 relevant elements)*

[✓] Import layout data only for elements that currently have no layout information


View layout configuration

Use a layouting algorithm

Select a layout algorithm for creating the layout data

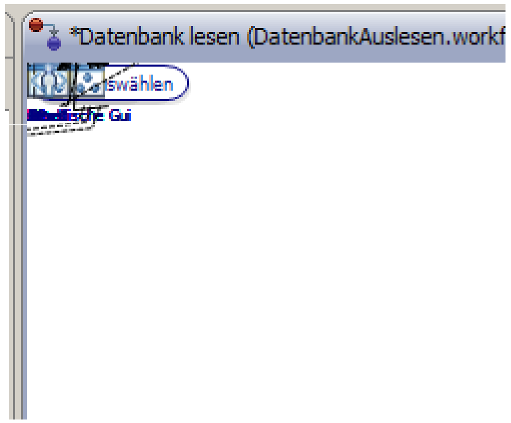
Set layout preferences for the view: ***Technical (Technical)***

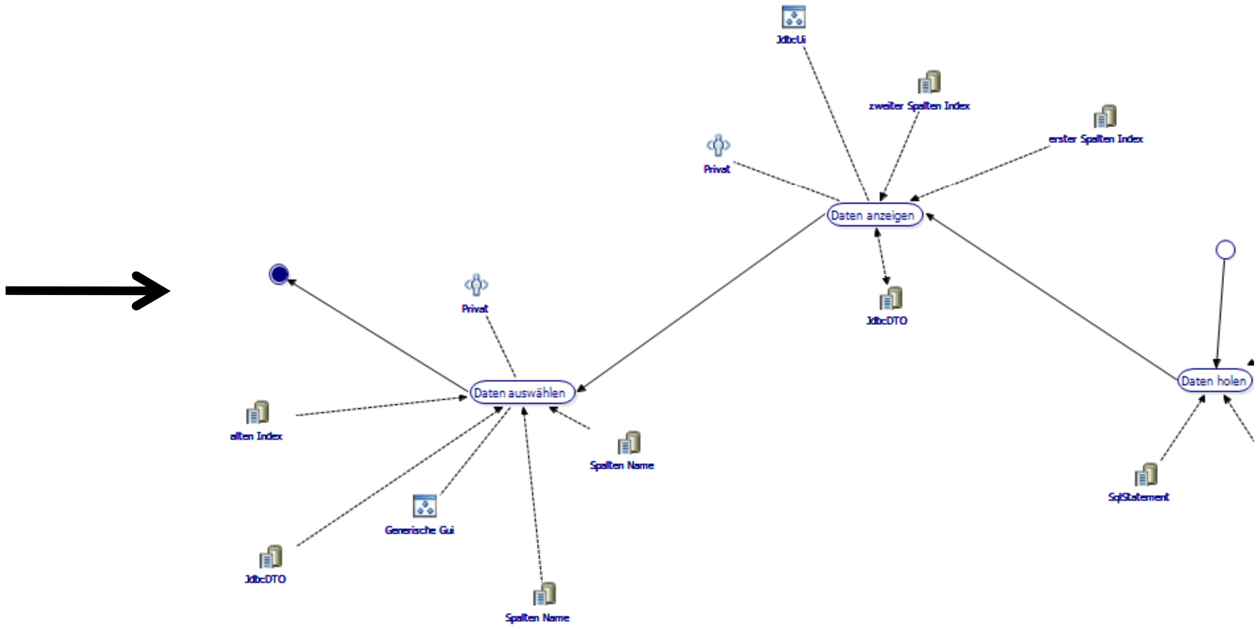
Please select a layouting algorithm: Spring

A spring layout.

Features – Workflow Editor

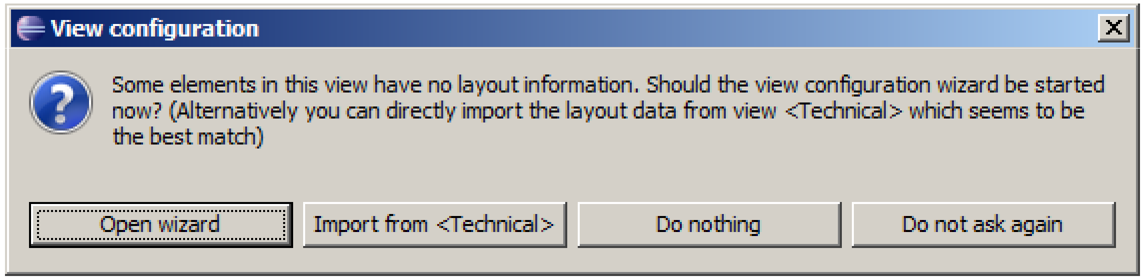
Applying automatic layout:





Features – Workflow Editor

When a view is opened that contains no layouting information, the user is automatically offered assistance

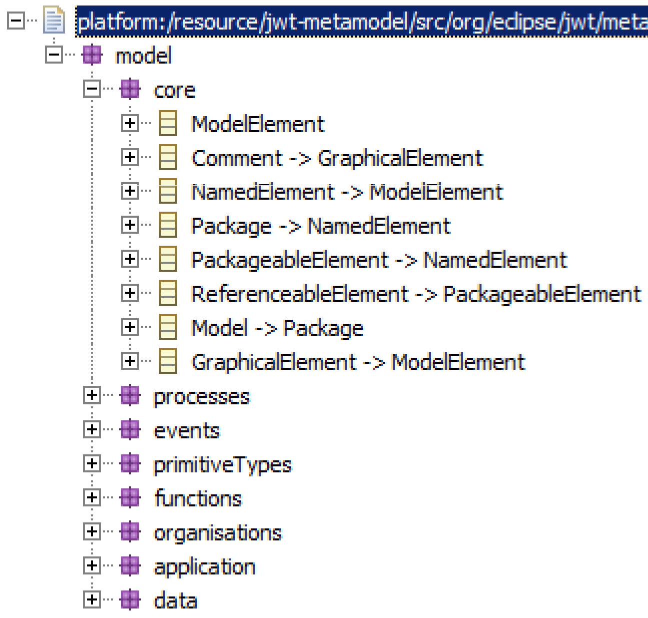


View configuration

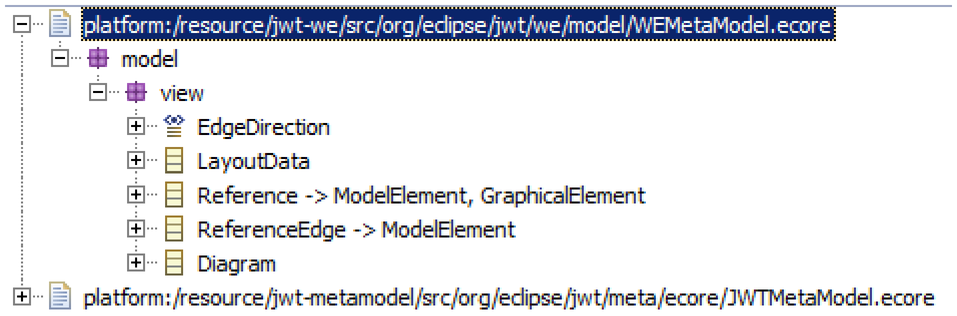
Some elements in this view have no layout information. Should the view configuration wizard be started now? (Alternatively you can directly import the layout data from view <Technical> which seems to be the best match)

Features – Workflow Editor

Separation of concerns:
The domain meta model is located in a plugin of its own, the WE contains a separate view meta model which extends the core model



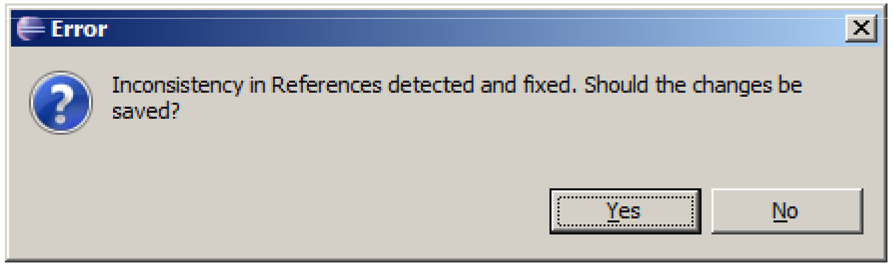
```
platform:/resource/jwt-metamodel/src/org/eclipse/jwt/meta/ecore/JWTMetaModel.ecore
  model
    core
      ModelElement
      Comment -> GraphicalElement
      NamedElement -> ModelElement
      Package -> NamedElement
      PackageableElement -> NamedElement
      ReferenceableElement -> PackageableElement
      Model -> Package
      GraphicalElement -> ModelElement
    processes
    events
    primitiveTypes
    functions
    organisations
    application
    data
```



```
platform:/resource/jwt-we/src/org/eclipse/jwt/we/model/WEMetaModel.ecore
  model
    view
      EdgeDirection
      LayoutData
      Reference -> ModelElement, GraphicalElement
      ReferenceEdge -> ModelElement
      Diagram
platform:/resource/jwt-metamodel/src/org/eclipse/jwt/meta/ecore/JWTMetaModel.ecore
```

Features – Workflow Editor

Robustness:
The Workflow Editor now has a fail-safe mechanism, that allows to restore broken view information. It can also be used to adapt workflow files which were modified outside the WE and therefore contain no view information.



Inconsistency in References detected and fixed. Should the changes be saved?

Features – Bug fixes

- Several minor bugfixes and enhancements, e.g. :
- A workflow model‘s companion conf model is also copied when using „Save as“
- Workflow models with unknown _conf extensions can still be opened and will show the extensions that are well known.
- _conf extension (Aspect) model default values are now used
- GDI memory leak in fonts/color resources has been fixed
- Missing/corrupted view information (References) is automatically corrected
- Now different external views do not cause problems with the FactoryRegistry

Non-Code Aspects - Documentation

- As everything else, documentation, communication and visibility is constantly being improved
- Several pages on the wiki describe to end users the usage of the Workflow Editor, Transformations, etc.
- Several pages on the wiki provide developer documentation. All extension possibilities by external plugins are documented there, and have examples that are available at least in the CVS.
- Such documentation is up to date.

Non-Code Aspects - Communication

- There has been a presentation about JWT at Eclipse Summit Europe 07, EclipseCon08, Open World Forum 2008, EclipseCon09, Solutions Linux 2009, Eclipse Forum Europe 2009 and Eclipse Europe Summit 2009.
- News have been posted on sites such as theserverside.com, JWT has now its project in Ohloh‘s directory, own project-related blog...
- In June 2009 a first publication appears in the German-speaking „Eclipse-Magazin“ describing the different features of JWT.
- In a follow-up article in the same magazine the view/aspect mechanisms of JWT have been presented
- NB. See about translation at UI Usability.

APIs

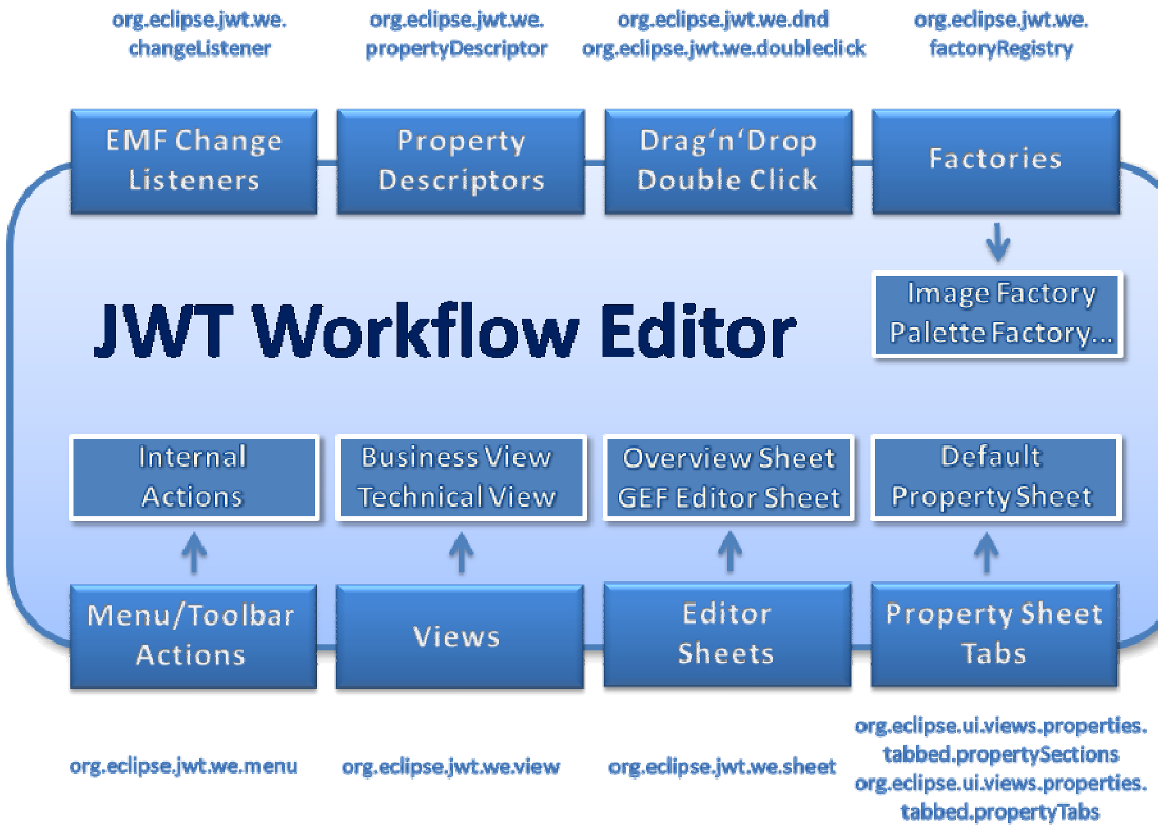
- The code conforms to the Eclipse Quality.
- The code has been implemented by committers of the project.
- The API has been officially declared (using PDE tooling) in prior to this release.
- Several examples have been tested and several jUnit tests exist.
- Build (includes unit testing) has been fully automated and is easily repeatable.

Architectural issues

- Workflow Editor
  - jwt-we: Contains Workflow Editor
  - jwt-view: Editor for Views
  - jwt-we-conf-model*: Custom aspect extensions for the meta-model
  - jwt-we-conf-property.*: Allows to add custom properties to model elements
  - jwt-we-*-sample: Several examples on how to use extension points
  - jwt-we-view-*: Additional views for the Workflow Editor
  - jwt-we-plugins: Several additions to the Workflow Editor
- Workflow Administration and Monitoring
  - jwt-wam-api: API to invoke a workflow service
  - jwt-wam-monitoring: Generic monitoring plugin

Architectural issues (cont.)

- Several extension points available that are already used by plugins in order to extend the JWT Workflow Editor.


org.eclipse.jwt.we.changeListener
org.eclipse.jwt.we.propertyDescriptor
org.eclipse.jwt.we.dnd
org.eclipse.jwt.we.doubleclick
org.eclipse.jwt.we.factoryRegistry
EMF Change Listeners
Property Descriptors
Drag'n'Drop Double Click
Factories
JWT Workflow Editor
Image Factory Palette Factory...
Internal Actions
Business View Technical View
Overview Sheet GEF Editor Sheet
Default Property Sheet
Menu/Toolbar Actions
Views
Editor Sheets
Property Sheet Tabs
org.eclipse.jwt.we.menu
org.eclipse.jwt.we.view
org.eclipse.jwt.we.sheet
org.eclipse.ui.views.properties.tabbed.propertySections
org.eclipse.ui.views.properties.tabbed.propertyTabs

Tool usability

- JWT enables a user to model his/her processes and workflows and use these models not only for documentation, but also for execution.
- The model can be used to generate code in different languages (such as XPDL)
- The model can be transformed to other models (e.g. STP BPMN).
- It already provides several extension points where others can build on it and extend several parts of the editor.
- If necessary, the meta-model can be adapted to the needs of each user.

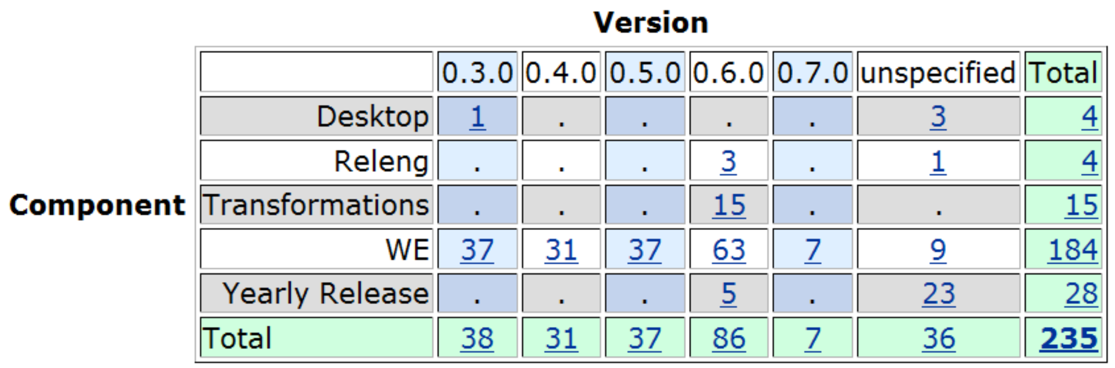
End-of-Life

- There are no features that are end-of-life‘d in this release.

Bugzilla

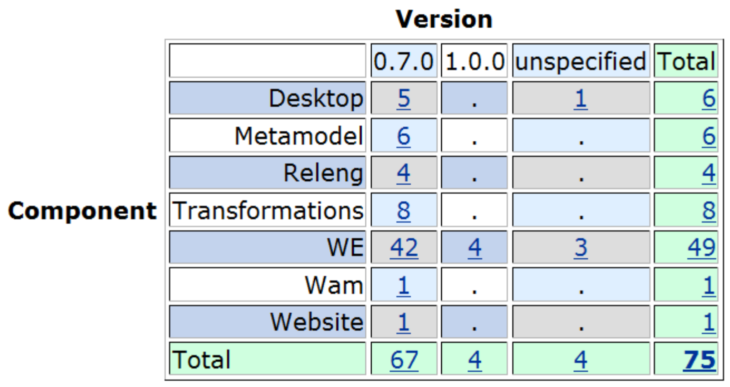
- Bugzilla currently knows 75 open bugs where most of them are feature requests for a future version of JWT (they will be moved to the next version in bugzilla)

Closed:



	Version						
Component	0.3.0	0.4.0	0.5.0	0.6.0	0.7.0	unspecified	Total
Desktop	1	.	.	.	.	3	4
Releng	.	.	.	3	.	1	4
Transformations	.	.	.	15	.	.	15
WE	37	31	37	63	7	9	184
Yearly Release	.	.	.	5	.	23	28
Total	38	31	37	86	7	36	**235**

Open:



	Version			
Component	0.7.0	1.0.0	unspecified	Total
Desktop	5	.	1	6
Metamodel	6	.	.	6
Releng	4	.	.	4
Transformations	8	.	.	8
WE	42	4	3	49
Wam	1	.	.	1
Website	1	.	.	1
Total	67	4	4	**75**

Standards

The relationship to existing standards has been achieved by the following:

  - Addition of a UML Activity diagram view
  - Addition of a EPC model view
  - Transformation to the Eclipse STP BPMN modeler
  - Codegeneration of XPDL possible

- Existing transformation to the STP-IM, which aims to bridge different standard oriented tools within the STP project, like BPMN and SCA editors or BPEL.

Standards (cont.)

- JWT has been enriched on the runtime side also:
  - Runtime process APIs allowing workflow engines in a standard way
    1. to integrate with any Java workflow engine
    2. To provide service orchestration features using any service platform. They have been contributed in the WAM and Runtime component.

  - The new OW2 Scarbo project provides its reference implementation, on top of the XPDL-compliant Bonita Engine and the SCA standard compliant Frascati service platform.
    - Because of licensing issues – in addition to Scarbo being a complete solution and not only tooling - this development has been contributed to OW2 as a new project called Scarbo, see http://scarbo.ow2.org/.

UI Usability

- The Workflow Editor supported originally several languages such as English, German and French. For the latest releases we collaborated with Babel so that much more languages now exist.
- Several wizards exist to ease the usage of the Workflow Editor.
- All transformations are easily selectable via Import and Export menus thanks to the transformation base.
- The Eclipse User Interface Guidelines have been followed.
- A UI walkthrough is planned for the following release

Schedule

- JWT 0.6 was successfull released along the Galileo yearly release.
- The original 0.7 release date of September 2009 has been postponed to December 2009.
- However, in addition to said new features, the past three months have been eventfull, with new users as well as new extenders (people developing a product on top of JWT), and fix of consequently reported bugs, all of it arguably part of Galileo‘s impact and worth releasing now.
- In 2010, JWT is going to move into the newly created SOA top level project. Together with the move review or shortly afterwards, we plan to graduate.

Communities

- JWT‘s mentor is John Graham (RedHat), former DTP Lead
- Active bugzilla usage by the committers and others (extenders).
- Many discussions on the mailing list, inside JWT as well as with partners from the STP projects (STP IM, BPMN, etc.).
- End user involvement has comparatively jumped up since Galileo. Often starts in the newsgroups, which is otherwise not used much.
- Discussions with several companies (Bull, jBoss) have produced collaboration (ex. BPM model comparison, FactoryRegistry) and keep feeding JWT‘s requirements.
- JWT Integrations include AgilPro and OW2 Scarbo (Open Source). Mailing list exchanges show new products being built on JWT.
- Coordination with several OW2 projects such as Bonita, Orchestra, FraSCAti, Spagic, Scarbo.
- Promising contacts with other teams following the Aspect Oriented Modeling talk at ESE 2009.
- The University of Augsburg is now an official Eclipse Foundation Member

IP Issues

- The code has been committed by individuals who are either committers of the project themselves or their foundation. Code contributions from others have undergone a CQ.
- The contribution questionaires for all components have been completed.
- The legal information has been inserted into the source code as described in the Eclipse IP Policy.
- The code has been approved by the EMO-IP-Team under CQ 1936, 2039, 2041, 2042, 2977, 3008, 3106, 3107, 3108, 3109 and 3110 as described on our IP Log http://www.eclipse.org/projects/ip_log.php?projectid=technology.jwt.
- There are no outstanding CQs and the IP log has been committed and approved for this release.

Project Plan

- We plan to move JWT to the upcoming SOA TLP, that after much discussions appears to all as our logical place, as soon as it exists. We will plan the move review accordingly.
- The next release will be scheduled afterwards as 0.8.0 or very possibly 1.0.0 (graduation release).
- It will consist of
  - new transformations (e.g. JWT to jPDL),
  - improved runtime and WAM component,
  - a first implementation of the desktop component as well as
  - minor bugfixes.
- URL
  - HTML: http://www.eclipse.org/projects/project-plan.php?projectid=technology.jwt
  - XML: http://www.eclipse.org/jwt/project-info/JWT_ProjectPlan_200909.xml

Communication Channel

- Committer Mailing List (preferred)
  - jwt-dev@eclipse.org
  - https://dev.eclipse.org/mailman/listinfo/jwt-dev
- Newsgroup
  - news://news.eclipse.org/eclipse.technology.jwt

Notes

- The Eclipse development process document and the Guidelines document have been read and approved by the project leads and committers of the JWT project.

Thanks for reading this document!

- Marc Dutoo (Project Co-Lead) - Open Wide, FR
- Florian Lautenbacher (Project Co-Lead) – University of Augsburg, DE

- John Graham (Mentor) – JBoss, a division of RedHat, US

- Christian Saad – University of Augsburg, DE
- Stéphane Drapeau – Obeo SA, FR
- Alain Boulze – INRIA / OW2, FR

- Miguel Valdez Faura, Mickaël Istria, Marc Blachon – Individuals, FR
- Guillaume Decarnin – Open Wide, FR
- Pierre Vigneras, Steve Egbert – Bull, FR
- Maxime Porhel – Obeo, FR
- and many others...

The JWT project team