

CloudScale Environment User Guide



Funded by the European Commission's
Framework Programme Seven (FP7).



Contents

<i>Installation</i>	5
<i>Requirements</i>	5
<i>Download</i>	5
<i>Installation & Run</i>	5
<i>Development</i>	6
<i>Maven (Tycho)</i>	6
<i>Eclipse IDE</i>	6
<i>Introduction</i>	7
<i>Environment perspective</i>	7
<i>CloudScale Environment Project</i>	7
<i>Dashboard</i>	8
<i>Workflow</i>	10
<i>Tools perspectives</i>	11
<i>Walkthrough</i>	12
<i>Examples</i>	12
<i>ScaleDL</i>	13
<i>Overview</i>	13
<i>Relation to Workflow</i>	13
<i>Features</i>	13
<i>Input & Output</i>	14
<i>Minimal Example</i>	14
<i>Extended Palladio Component Model (Extended PCM)</i>	15
<i>Relation to Workflow</i>	15
<i>Features</i>	15
<i>Input & Output</i>	15
<i>Minimal Example</i>	15
<i>Architectural Templates (ATs)</i>	17
<i>Relation to Workflow</i>	17
<i>Features</i>	17

<i>Input & Output</i>	17
<i>Walkthrough</i>	Error! Bookmark not defined.
<i>Usage Evolution (UE)</i>	19
<i>Relation to Workflow</i>	19
<i>Input & Output</i>	19
<i>Walkthrough</i>	19
<i>Extractor</i>	24
<i>Introduction</i>	24
<i>Relation to Workflow</i>	24
<i>Problem</i>	24
<i>Features</i>	25
<i>Input & Output</i>	25
<i>Walkthrough</i>	25
<i>References</i>	26
<i>Analyser</i>	27
<i>Introduction</i>	27
<i>Relation to Workflow</i>	27
<i>Problem</i>	27
<i>Features</i>	28
<i>Input & Output</i>	28
<i>Walkthrough</i>	28
<i>References</i>	29
<i>Static Spotter</i>	30
<i>Introduction</i>	30
<i>Relation to workflow</i>	30
<i>Problem</i>	30
<i>Features</i>	30
<i>Input & Output</i>	31
<i>Walkthrough</i>	31
<i>Dynamic Spotter</i>	33
<i>Introduction</i>	33

<i>Reference to workflow</i>	33
<i>Problem</i>	33
<i>Features</i>	34
<i>Walkthrough</i>	34

Installation

Requirements

- *Java 7 (JRE is not yet included)*
- *Windows, OSX or Linux operating system (32 or 64 bits)*
 - *If 32-bit Java is used on 64-bit OS, 32-bit bundle needs to be downloaded*

Download

- *CSE bundles available at <http://www.cloudscale-project.eu/results/tools/>*
 - *OS is automatically detected*
 - *Choose between Release and Nightly version*

Installation & Run

1. *Download bundle*
2. *Extract/Unzip bundle*
3. *Open folder and run Environment (eclipse)*

Development

Maven (Tycho)

1. Clone the repository.
 - `$ git clone https://github.com/CloudScale-Project/Environment.git`
2. Build Cloudscale Environment.
 - `$ mvn package`
3. Run Linux, MacOS, Windows distribution.
 - Bundle location: `plugins/eu.cloudscaleproject.env.master/target/products`

Eclipse IDE

1. Download and install Eclipse Luna for RCP and RAP
2. Download and install Eclipse plugin dependencies for CloudScale development
 - a. Go to Eclipse->Help->Install New Software
 - b. Add CloudScale Toolchain update site:
<http://cloudscale.xlab.si/cse/updatesites/toolchain/nightly/>
 - c. Install Toolchain features (Analyser, Extractor, Static Spotter and Dynamic Spotter), Dependencies and Sources (sometimes Dependencies needs to be installed first and after that everything else)
3. Clone repository
 - a. `$ git clone https://github.com/CloudScale-Project/Environment.git`
4. Import CloudScale Environment plugins, under "plugins/" directory, into the workbench.
5. Run product (`eu.cloudscale.env.product`)

Introduction

The CloudScale Environment (CSE) is an open-source solution oriented to provide an engineering approach for building scalable cloud applications by enabling analysis of scalability of basic and composed services in the cloud. It is a desktop application integrating CloudScale tool-chain, consisting of Dynamic and Static Spotters, the Analyzer and the Extractor, while driving the user through the flow of the CloudScale Model. Application can be installed and used in any personal computer running Java 6+, including Windows, MacOS and Linux.

Environment perspective

Environment perspective is the main perspective in the CSE, responsible to provide all main functionality of the CloudScale toolchain through unified and well defined components. It can be accessed through Tools>Environment>Perspective action or through toolbar action (first button). In this perspective of the tools' actions and views are hidden since all functionality is available through integration components. To use Tools in a stand-alone fashion see 'Tools perspectives' section.

CloudScale Environment Project

The initial step in using the CloudScale Environment is creation of a project. A project represents a system being analyzed by the tool user. As showed in Figure 1, a project initially contains project specific files, dedicated ScaleDL models folder with automatically generated ScaleDL Overview model and diagram, and dedicated tool's folders.

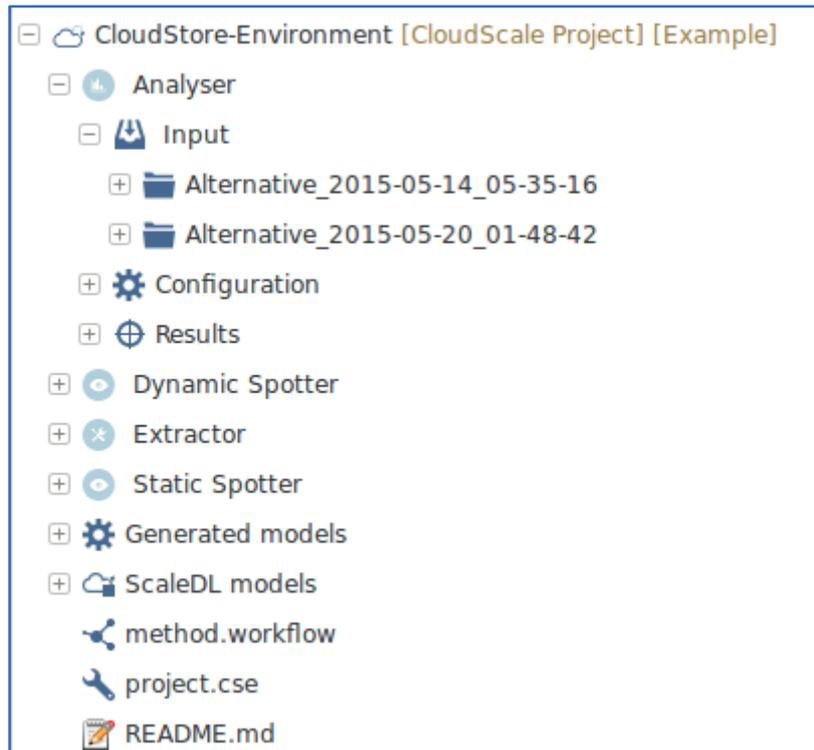


Figure 1: CloudScale Environment project

Project specific files are *project.cse* and *method.workflow*. First contains general information about the project (location of files, results, etc.) and is presented through the Project Dashboard. The *method.workflow*, on the other hand stores the current state of the project (models validations, enabled and disabled steps, etc.) which are presented through the workflow diagram. *ScaleDL models* folder contains, as name suggests, all *ScaleDL* models (Overview, Usage Evolution and Architectural Templates) and corresponding diagrams. Additionally, all PCM models imported into an Overview model are stored in the “imported” subfolder. For each integrated tool, corresponding plugin creates folder where it stores all tool specific information. In general all tools needs input, configuration and results, therefore these are also present in first level in subfolder. Further down it depends on specific integration which data is stored and how the data is organized. Nevertheless, users can always find all tool’s specific information (e.g. models, configurations, results, etc.) inside these folders.

Dashboard

To provide unified access to all the integrated tools, project Dashboard component has been developed. In general each integrated tool needs some input, run configuration and at the end it provides some results. Taking this into account, dashboard contains main user-interface component for each tool, which is further divided into three sections: input, configuration, results.

Input section defines what the input is (e.g. complete PCM model for Analyser) and how to acquire it (e.g. output of the Overview transformation is input into the Analyser). Supporting different CloudScale Method paths, one can use the tool independently (i.e. define manual input) or connect other tool's output to its input. In the current state the integration mechanism does not yet support alternative inputs, which is planned for the next year.

Configuration section defines what configurations are required and provides option to run the tool. Since all integrated tools contain their own run configuration user interfaces, the dashboard does not try to duplicate it, however it tries to provide easier access to the configurations, through pre-configuring what is possible, giving users options to create, delete and modify configurations and run them.

Results section displays all results from tool runs. Since the tools also provide their own user interfaces for displaying results, the results component shows which results are available, status of the run and provide an option to open specific result in dedicated component. Nevertheless the dashboard is currently used only for integrated tools, it also provides the extension point, where additional plugins can extend its functionality by providing new tools and/or operations.

- **Analyser**
 - **Input** - Input models (PCM) with addition to automatically create new ones or import existing ones and apply Architectural Templates.
 - **Run** - convenient manipulation of the Experiments model and running capabilities, supporting Scalability/Capacity and Normal run configurations
 - **Results** - persist entire result sets and shows most important informations (including charts), with possibility to open dedicated EDP2 views.
- **Extractor**
 - **Input** - currently not visible, since input can for now only be already imported Java projects.
 - **Run** - automatically configures Modisco and SoMoX targets based on the project selected. It also exposes metrics used in extraction.
 - **Results** - all extraction data are persisted in results folder and showed through this component; extracted PCM models, modisco models, ...
- **Static Spotter**
 - **Input** - Source Decorator model produced by SoMoX with corresponding PCM models.
 - **Run** - view and manipulation with Static Spotter catalog. Run static analysis.
 - **Results** - persists and displays all anti-patterns found in the results.
- **Dynamic Spotter**
 - **Server** - configuration of Spotter server. In CSE user can work with integrated one or he can configure address of external spotter server.

- **Input** - Provides Instrumentations and Measurements configurations.
- **Run** - Provides basic Dynamic Spotter configurations with Workflow and Hierarchy selection
- **Results** - all results are persisted in results folder and showed through this component.

Workflow

The workflow diagram represents all possible paths that user can take in the CloudScale Environment to analyse scalability of the system.

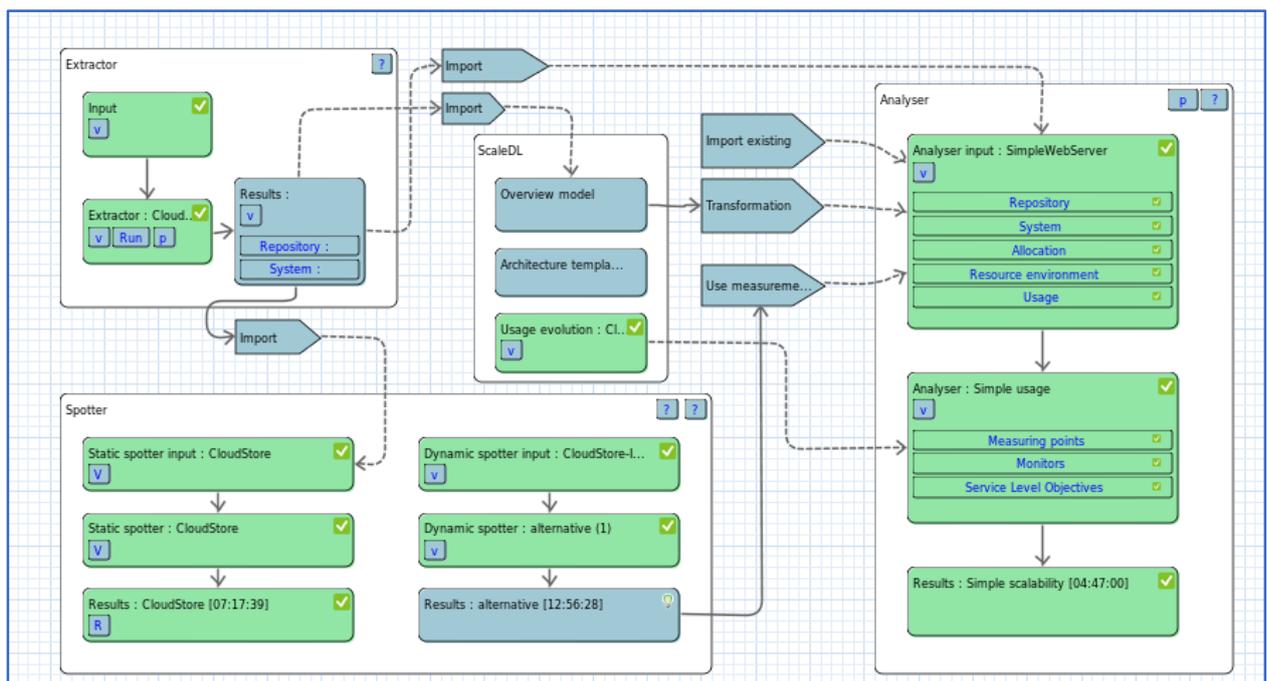


Figure 2: Workflow diagram (with successfully validated alternatives)

The diagram itself is composed of 4 main groups; Analyser, Extractor, Spotter and ScaleDL. Each group defines a set of states and how they are depended on each other, while external boxes represents actions (i.e. generate, import, use) that can be executed to connect results of one tool to the input of another. The tools groups contain three states; input, configuration and results, and these are analogous with the dashboard sections; which are displayed by double-clicking on the state. In addition these states also contain a toolbar with available actions (e.g. wizards, quick run) and a representation of; required resources in an input state, available run configurations in a run state and available results in a results state.

Workflow diagram has two types of connections; required (solid) and optional (dashed). First defines that current state is active only when all required states are validated, while the latter one does not expect the optional state to be validated, however it normally represents tools interoperability and thus producing required resources quicker. Entry points, where

user can start at the beginning are all the states that does not require another one to be fulfilled; tools' input states and all ScaleDL models.

Tools perspectives

Complete CloudScale tool-chain is available in the CloudScale Environment separately through dedicated tool's perspectives, which can be accessed through Tools menu or directly using toolbar buttons. From these perspectives tools can be used separately as a stand-alone products without the need of using Environment integration functionalities. For instructions how to use integrated tools separately, please see Tools tutorials.

To open other views (UI components) that may be missing in the menus, one can access them through "Tools>Show View" action (Alt+Shift+Q Q).

The CloudScale Environment is designed to be fully extensible, giving the possibility to the open-source community of improving and adapting it to its own needs. All the integrated tools (Palladio, SoMox, Reclipse and the SoPeCo) will be available from within the application through separate perspectives; however the main perspective (CloudScale perspective) will try to cover as much functionality as needed through dashboard and workflow components (see 4.2) to achieve a seamless and integrated user experience providing functionalities described before.

- **Analyser (Palladio)** - Palladio is a software architecture simulation framework, which analyses software at the model level for performance bottlenecks, scalability issues and reliability threats, and allows for a subsequent optimisation. In the CloudScale Environment mainly the backend engines will be used to analyse the user's application. This will be achieved by automatically transforming ScaleDL Overview models into the Palladio Component Models; structure on which Palladio operates.
- **Extractor (SoMoX)** - SoMoX provides clustering-based architecture reconstruction to recover the architecture of a software system from source code. The clustering mechanism extracts a software architecture based on source code metrics and construct PCM models to be used by Analyser.
- **Dynamic and Static Spotter** Spotter is a tool for identifying ("spotting") scalability problems in Cloud applications. The tool consists of two main components that are provided as two separate programs: Static and Dynamic spotters. The Static Spotter analyses the application code to find scalability anti-patterns. The Dynamic Spotter analyse the execution of the application (through systematic measurements) to find scalability anti-patterns in the application's behaviour. Within this document we will refer to the pair as Spotter, and to their individual components as either Dynamic or Static Spotter.

Walkthrough

Below are steps and most basic information needed to start working with the CloudScale Environment.

1. *Start CloudScale Environment*
2. *Open Environment perspective (through menu Tools>Environment>Perspective)*
3. *Create new CloudScale Project*
 - a. *File->New->CloudScale Project*
 - b. *Click on first toolbar item ('New CloudScale project')*
4. *Go through available components*
 - a. *Dashboard accessible through double-clicking 'project.cse' file*
 - b. *Workflow accessible through double-clicking 'method.workflow' file*
 - c. *ScaleDL models available under 'ScaleDL models' folder*
 - d. *Each Tool has has dedicated folder where all configurations and results are stored*
5. *Start creating input and run alternatives for the tool of your choice using Dashboard components. Help with Workflow diagram which highlights steps that are available and shows validation information (if everything is ok or something needs to be done/fixe).*

Examples

CloudScale Environment integrates many examples and templates including stand-alone examples for each of the integrated tools. Examples can be accessed through CloudScale Examples project in New Project wizard.

ScaleDL

The Scalability Description Language (ScaleDL) is a language to characterize cloud-based systems, with a focus on scalability properties. ScaleDL consists of five sub-languages: three new languages (ScaleDL Usage Evolution, ScaleDL Architectural Template, and ScaleDL Overview) and two reused language (Palladio's PCM extended by SimuLizar's self-adaption language and Descartes Load Intensity Model (DLIM)). In this section for each we describe how it is used in the CloudScald Environment.

Overview

The ScaleDL Overview is a meta-model that provides a design-oriented modelling language for cloud-based system architectures and deployments. It has been designed with the purpose of modelling cloud based systems from the perspectives of service deployments, dependencies, performance, costs and usage. It provides the possibility of modelling private, public and hybrid cloud solutions, as well as systems running on non-elastic infrastructures.

Relation to Workflow

An instance of the Overview model is a part of a ScaleDL instance. It can be generated from the Extractor result, partial PCM model, or directly by creating a new Overview alternative in the Dashboard editor. Overview model, when completed, can later be used to produce the input model for the Analyser (see [Extended Palladio Component Model](#)).

Features

An Overview model instance allows software architects to model platform services, software services, connections between them and deployment in the cloud environment. It includes hardware specifications for the different cloud providers. Those specifications are designed in a way to support extending existing cloud specifications, by the use of a system descriptors.

Overview meta-model can be used as a general structure for describing software architecture inside cloud environment.

In CloudScale Environment its main purpose is to ease and accelerate the modeling of the Extended PCM model, by the use of partial PCM to Overview and Overview to Extended PCM transformations. Software services in Overview model can embed partial PCM models to describe internal mechanics. When the Overview model is transformed to the Extended PCM, partial PCM models are combined to form a complete input for the Analyser.

Input & Output

Overview model can be modelled with the Overview diagram editor. Initial configuration can be made with the import wizard from the Extractor output model, or external partial PCM model (Repository and System).

The main output of the Overview model is the Extended PCM model, which is used as the input for the Analyser (see [Analyser](#)).

Minimal Example

To create an Overview model instance, user has to create a new Overview alternative. This can be achieved by double clicking on the Overview section in the Workflow diagram, or by using the Dashboard editor. In the Dashboard editor, user interface section for manipulating Overview alternatives is under the Overview tab item. Create button opens up the wizard for creating an empty Overview alternative, or creating initial model from the Extractor result or external partial PCM model.

Extended Palladio Component Model (Extended PCM)

Extended PCM allows architects to model the internals of the services: components, components' assembly to a system, hardware resources, and components' allocation to these resources, the extension allows additionally to model self-adaptation: monitoring specifications and adaptation rules.

Relation to Workflow

An instance of the Extended Palladio Component Model (Extended PCM) is part of a ScaleDL instance, the input of the Analyzer (see [Workflow](#)).

Features

A PCM instance allows software architects to model the internals of services. The PCM allows to model components, components' assembly to a system, hardware resources, components' allocation to these resources, and static usage scenarios.

The “extended” refers to additional models added in the CloudScale context. These models cover monitoring specifications, service level objectives, and self-adaptation rules. Monitoring specifications allow to mark PCM elements, e.g., an operation of a component, to be monitored during analysis using a metric such as response time. Service level objectives specify thresholds for these metrics, allowing software architects to manifest their quality-related requirements. Self-adaptation rules can react on changes of monitored values. For example, when a certain response time threshold is exceeded, an adaptation rule could trigger a scaling out of bottleneck components.

Input & Output

Creating Extended PCM instances is mainly based on the input of the software architects that create these instances. Software architects can, however, be supported by using Architectural Templates (see Architectural Templates (ATs)) or by reverse engineering partial Extended PCM instances from source code (see Extractor).

The main output for specifying an Extended PCM is the Extended PCM instance itself. It can be used for documenting a system's architecture and is part of a ScaleDL instance; the Analyser input (see Analyser).

Minimal Example

To create an Extended PCM instance, software architects conduct the following actions:

1. Switch to the Analyser Perspective (second button of the image below).



2. *Create an Extended PCM instance by following the Palladio Workshop white paper and the Analyser screencast series. For example, PCM component repositories are created via the  button shown in the image above.*

Architectural Templates (ATs)

ScaleDL Architectural Template allows architects to model systems based on best practices as well as to reuse scalability models specified by architectural template engineers.

Relation to Workflow

Architectural Templates (ATs) are part of ScaleDL the input of the Analyzer (see Workflow). Also the transformation from “Overview model” to “Analyzer input” utilizes ATs to simplify the transformation specification.

Features

Architectural Templates (ATs) help software architects to specify Extended PCM instances more efficiently based on reusable ScaleDL templates. We provide the CloudScale AT catalogue that includes best practice templates for designing and analyzing scalable, elastic, and cost-efficient SaaS applications. We based this catalogue on common architectural styles and architectural patterns found in cloud computing environments.

Input & Output

For an AT-based design, software architects need a catalogue of available ATs as an input. For cloud computing applications, we suggest the CloudScale AT catalogue.

The output of an AT-based design is part of a normal ScaleDL instance. The Analyser fully supports AT-enabled ScaleDL instances.

Walkthrough

Software architects conduct the following actions for designing ScaleDL instances with ATs:

1. Switch either to the “CloudScale Perspective” or the “Analyser Perspective” using the  menu bar buttons.
2. Open an existing PCM system model or created a new one (see Extended Palladio Component Model (Extended PCM)).
3. Open the system in a tree editor (i.e., double-click the *.system file, not a *.system_diagram file).
4. Right-click on the “system” element of your system model and choose “MDS D Profiles... -> Apply Profile”.
5. In the new dialog, select the AT you want to apply and apply it. By default, all ATs of the CloudScale AT catalogue are available. If you want to use third-party AT catalogues, assure that you installed the respective plugins; included ATs will then also be shown in this dialog.

6. *Assign all AT roles to respective elements of your ScaleDL instance. For example, if an AT specifies an AT role for PCM's assembly contexts, right-click this assembly context within the system model tree editor and select "MDS D Profiles... -> Apply Stereotype" and apply the role similar to the previous action. Our CloudScale AT catalogue describes, per AT, which roles have to be assigned.*
7. *As each AT role can have a set parameters, software architects also need to set such parameters when requested. These parameters can simply be set via the Properties View for elements that have an AT role assigned.*

Usage Evolution (UE)

The purpose of the ScaleDL Usage Evolution is to enable modelling of how the load and work of a system evolves over time. ScaleDL Usage Evolution provides a thin modelling layer that expresses how the evolution of a set of variables applies to a usage model expressed in the Palladio Component Model (PCM). The actual modelling of the evolution of a single variable (e.g. arrival rates or population in load or average video size in work) is done using the LIMBO Load Intensity Modelling Tool. The details of using the Palladio and LIMBO are not covered here, but we refer instead to the PCM chapter of this document, the Usage Evolution screencast, and to the tutorials and screencasts on the Palladio home pages.

Relation to Workflow

An instance of the Usage Evolution is part of a ScaleDL instance and defined under the ScaleDL step in the workflow. It is an input of the Analyzer (see Workflow).

Input & Output

Walkthrough

The following is an overview of the steps for creating and running a usage evolution simulation:

1. Create a Palladio Component Model (PCM) of the system to evolve, including a Usage model with at least one Usage Scenario.
2. Create LIMBO models for how load and selected work parameters (e.g. amount of data to process) evolves time.
3. Create a Usage Evolution model that maps how LIMBO models are used to describe the evolution of load and work parameters from a selected usage scenario.
4. Create a run configuration
5. Run the simulation (see details under the Analyser section).
6. (optional) Adding additional usage and evolution models

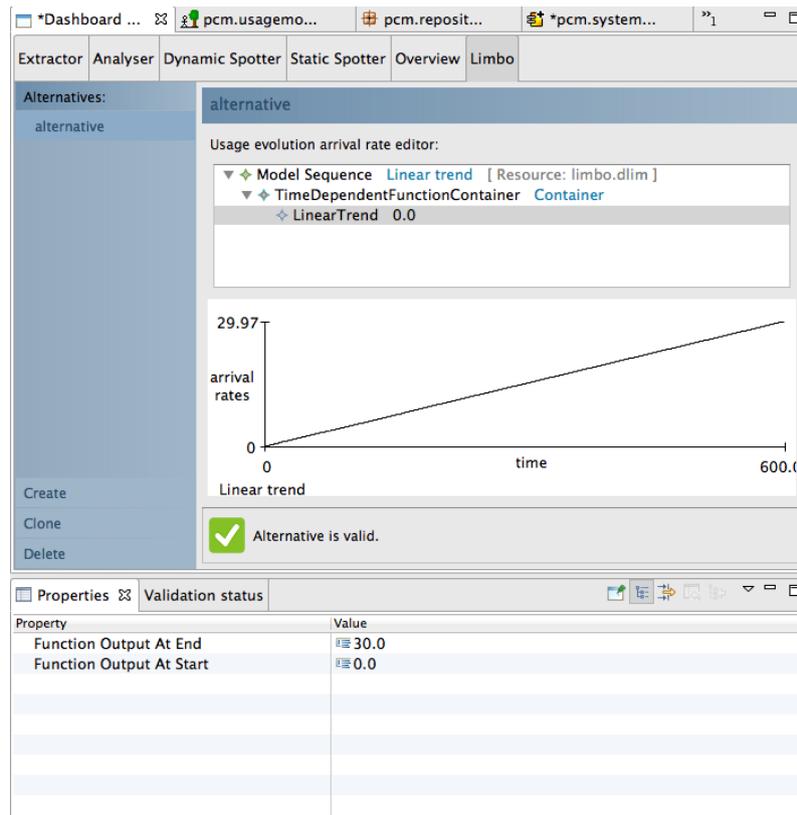
Each of the steps are described in more detail in the sub-sections below.

Creating the PCM model

The first step is to create or import a Palladio Component Model (PCM) of the system to evolve, including a Usage model with at least one Usage Scenario. This task can e.g. be initialised from the Analyser tab of the Dashboard, where the set of PCM models can be created or imported by selecting Input, pressing the “Create new...” button and selecting one of the input alternatives. See the Analyser and PCM chapter for details on this.

Creating LIMBO models

The next step is to create LIMBO model of how the load and selected work parameters from the PCM model evolve over time. LIMBO models can be created from the LIMBO tab of the Dashboard. From the dialog that appears when selecting to create a LIMBO model, it is possible to create either an empty LIMBO model, a model based on a template, or to select a more advanced setup through a wizard. Note that the wizard interface cannot be used to edit the model after it has first been created.



The LIMBO tool provides a tree-based editor as shown in the figure. At the root of the model is a Sequence which typically contains one or more TimeDependentFunctionContainer that can be regarded as segments of the sequence. Each segment has a duration that adds to the total duration of the Sequence. When using multiple LIMBO models (to describe work parameters as well as load) the total duration of each Sequence must be the same.

Each segment contains a single function (e.g. linear, exponential, logarithmic or sin trend) describing how the value evolves within that segment, and typically contains a start and end value (see e.g. properties for LinearTrend in the figure).

Note that the LIMBO tool originally was developed with focus on arrival time, and thus uses "arrival time" in some of the labels (e.g. the y-axis in the graph). However the same modeling concept can be use to express the evolution over time of other values such as

population in closed workload or even work parameters such as average data size (e.g. video or picture size).

See the LIMBO web pages for more documentation on the tool.

Creating a run configuration

The next logical step is to create a usage evolution model, but as this model is initialised along with the run configuration, we describe creating a run configuration first.

To create a run configuration, open to the Analyser tab of the Dashboard and the Run tab within it. Click the create button in the “Alternative conf:” column. When no configuration has been added yet, you can also use the “Create new...” button that will be present in the main editing area on the middle/right side of the screen.

For the usage evolution, a “Normal” alternative type should be selected in the dialog that appears. Under basic settings, measurement count stop condition should be unchecked (set to “-1”). The value of simulation time stop condition should to equal or higher than the Final Duration of the LIMBO models.

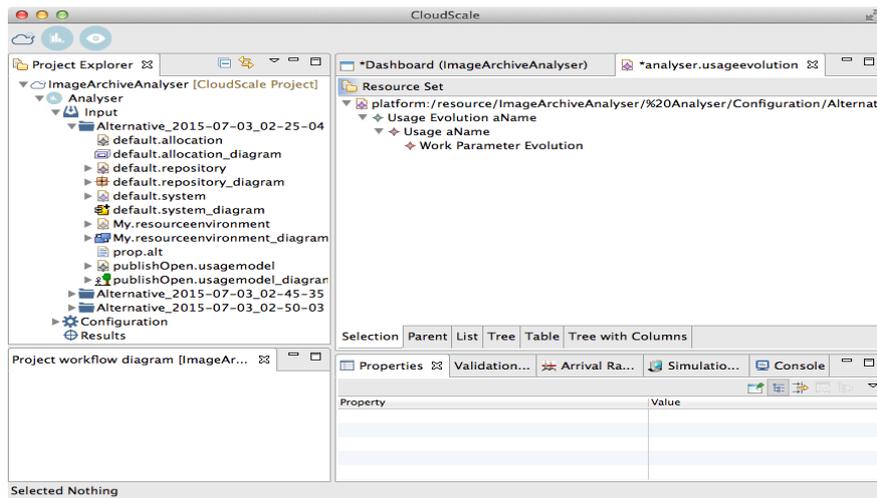
The details of setting up a run configuration is further described in the Analyser chapter.

Creating the Usage Evolution model

The usage evolution model describes how a usage scenario of a usage model evolves over time, and uses LIMBO models to define the evolution of the individual load and work parameters.

As described under Creating a run configuration, a usage evolution model is created as part of a run configuration, and will appear under the subfolder of your selected configuration alternative for the analyser.

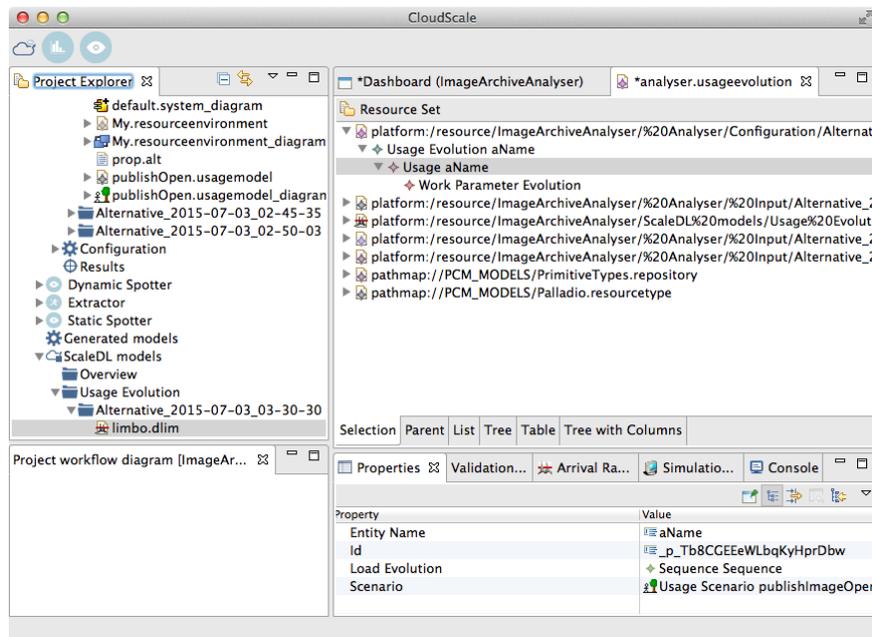
The figure below shows the ScaleDL Usage Evolution tree-based structure editor. At the root of the model is a single usage evolution element. To add content to the model, select the root element, and select “New Child / Usage” from the context menu to create a Usage. Then, select the newly created Usage element - its properties will be shown in the Property editor tab at the bottom of the screen.



To reference elements from the usage model and LIMBO models created in previous steps in the usage evolution, they must first be imported into the usage evolution editor. This is done by dragging the files into the editing area of the usage evolution editor (or use “Load resource...” from the context menu to locate them). The usage model file is found under the subfolder of your selected input alternative for the analyser, while the LIMBO models are found under the LIMBO subfolder of the ScaleDL models.

Once the files have been loaded, make sure the Usage element in the editor is selected. Click on the Load evolution property, and select the sequence from the LIMBO model that appears in the list. Then, click on the Scenario property and select the usage scenario from Palladio that appears in the list. The editor should now look similar to what is shown in the next figure. Remember to save the model once the edits have been completed.

If there is any work parameters in your model that you want to describe evolution of, add one Work Parameter Evolution childs to the Usage for each of them. From the Evolution property select the LIMBO sequence that describe the work parameter evolution (remember that each LIMBO model must first be first be imported into your model). Set the Variable Characterisation property to point to the variable from the PCM model to evolve.



Running the simulation

Open the Run tab under the Analyser in the Dashboard. Select the configuration you want to run, and press the Run button at the bottom right of the screen. To see the result, first check that you are in the Analyser perspective (click the graph-icon in the top left of the window). The results of the simulation appears in the Experiments view that is shown in the tab next to the Project Explorer in the Analyser perspective. See the [Analyser](#) section for more details on running simulations.

Adding additional usage and evolution models

In some cases it is will be useful to create more than one usage scenario and/or usage evolution description to analyse different uses of the systems or how it will evolve in different future scenarios. To add this, from the Input tab for the Analyser tab in the dashboard:

- Click the create button under the “alternative inputs column” to create an additional analyser input alternative. In the wizard that appears, import the models with exception of the usage model from the initial alternative to them.
- Create a new usage model by clicking the “Create...” button on the right hand side of the Analyser tab, and select Usage Model and Usage Evolution Model.
- Add a new run configuration, where the Usage Evolution for the alternative will be located.
- Add content to the new models
- As required, import or create new Limbo models to describe the evolution of individual parameters, and refer to these from the usage evolution mode as described in the previous section.

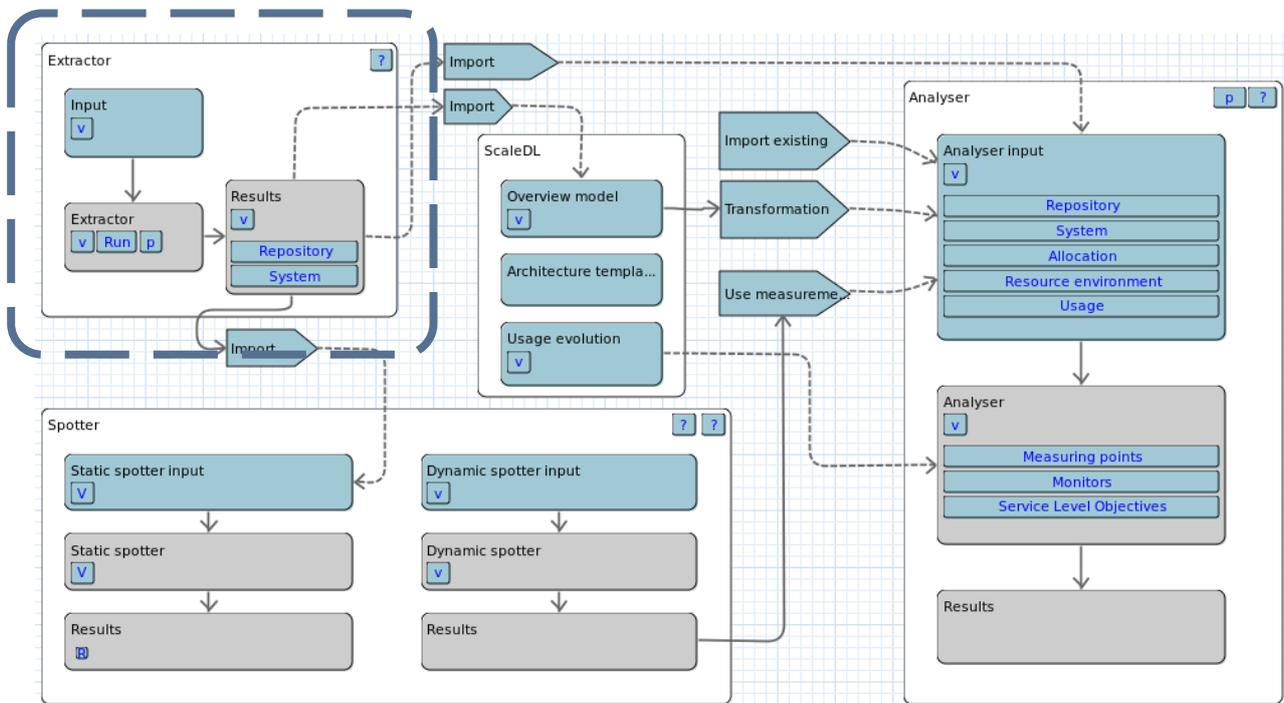
Extractor

Introduction

The Extractor is a reverse engineering tool for automatic model extraction. It parses source code and generates partial ScaleDL models that are further used by the Analyser.

Relation to Workflow

The Extractor is a reverse engineering tool for automatic model extraction. It parses source code and generates partial ScaleDL models that are further used by the Analyser and the Static Spotter.



Problem

There are situations for a software system that the original architecture has been lost or outdated due to further development. The software architect has no detailed picture of what the software system's architecture looks like but only the source code. While the software architect would require the architecture when migrating the software system to a new environment such as cloud computing environment.

Features

The Extractor is based on the Archimatrix approach. Archimatrix is a tool-supported reengineering process that combines different reverse engineering approaches to enable an iterative recovery and reengineering of component-based software architectures. The Extractor is focused on parsing and architecture reconstruction. The Extractor parses the source code, extracts and clusters a component-based software architecture based on source code metrics.

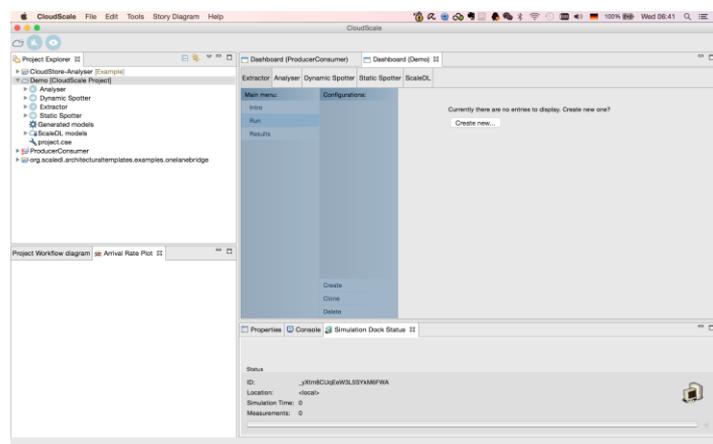
The software architecture is generated as a partial ScaleDL model.

Input & Output

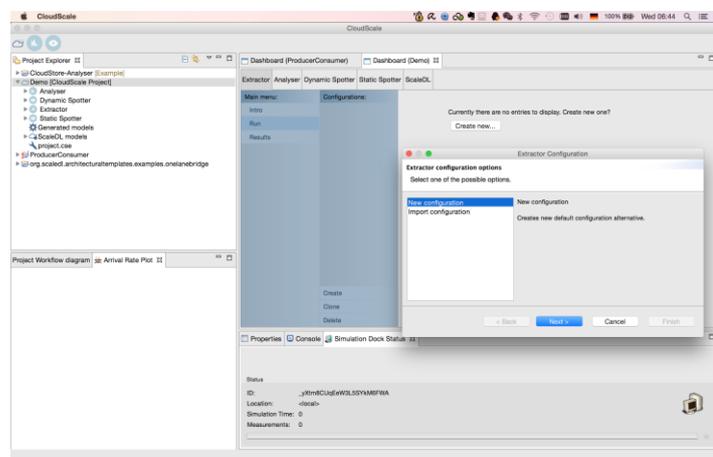
As for an input, the Extractor takes Java source code and generates partial ScaleDL model as an output

Walkthrough

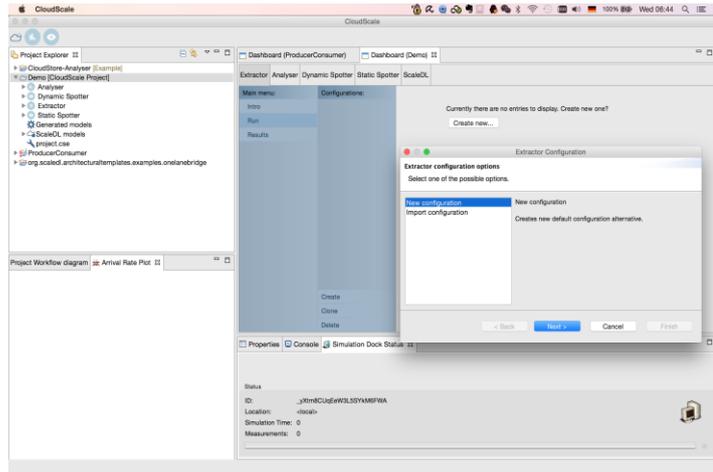
1. Click the Extractor on Dashboard, go to Run



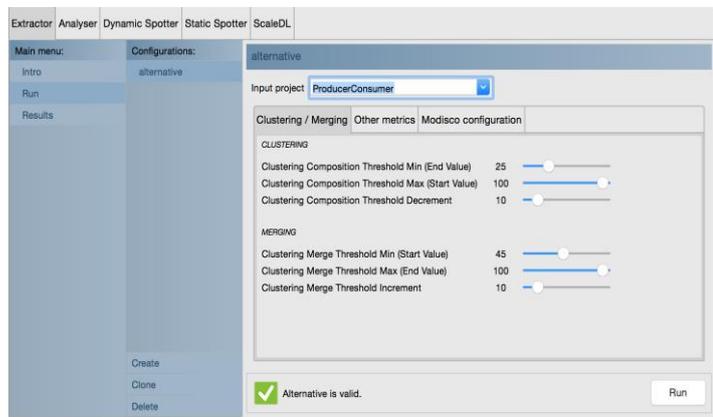
2. Click Create, Create a new configuration



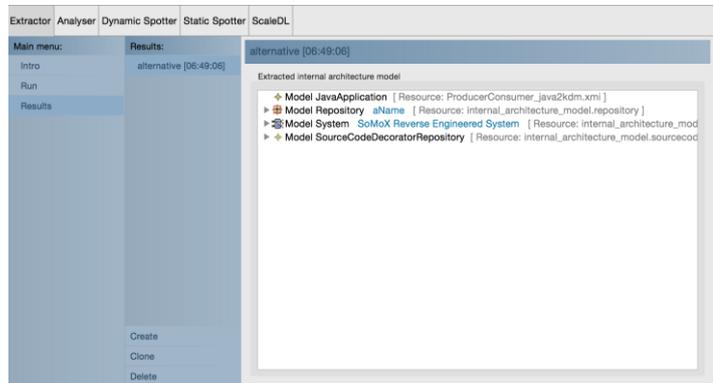
3. *Select input project, configure the metrics*



4. *Run*



5. *Click Results tab under Main menu, you can see a bunch of models have been generated.*



References

Ref. to CS Method

Ref. to Extractor User manual

Analyser

Introduction

CloudScale's Analyser allows to analyze ScaledL models regarding scalability, elasticity, and efficiency of cloud computing applications at design time. For these capabilities, CloudScale integrated novel metrics for such properties into Analyser. Analyses are based on analytical solvers and/or or simulations. Analyser particularly supports to analyze self-adaptive systems, e.g., systems that can dynamically scale out and in.

Relation to Workflow

The Analyser is important for the right-hand side of the Workflow (see image below). It is capable of providing scalability, elasticity, and cost-efficiency analysis results based on a fully specified ScaledL instance. Such analysis results can already be provided before the system is accordingly implemented, e.g., at early design time or to investigate modification scenarios.

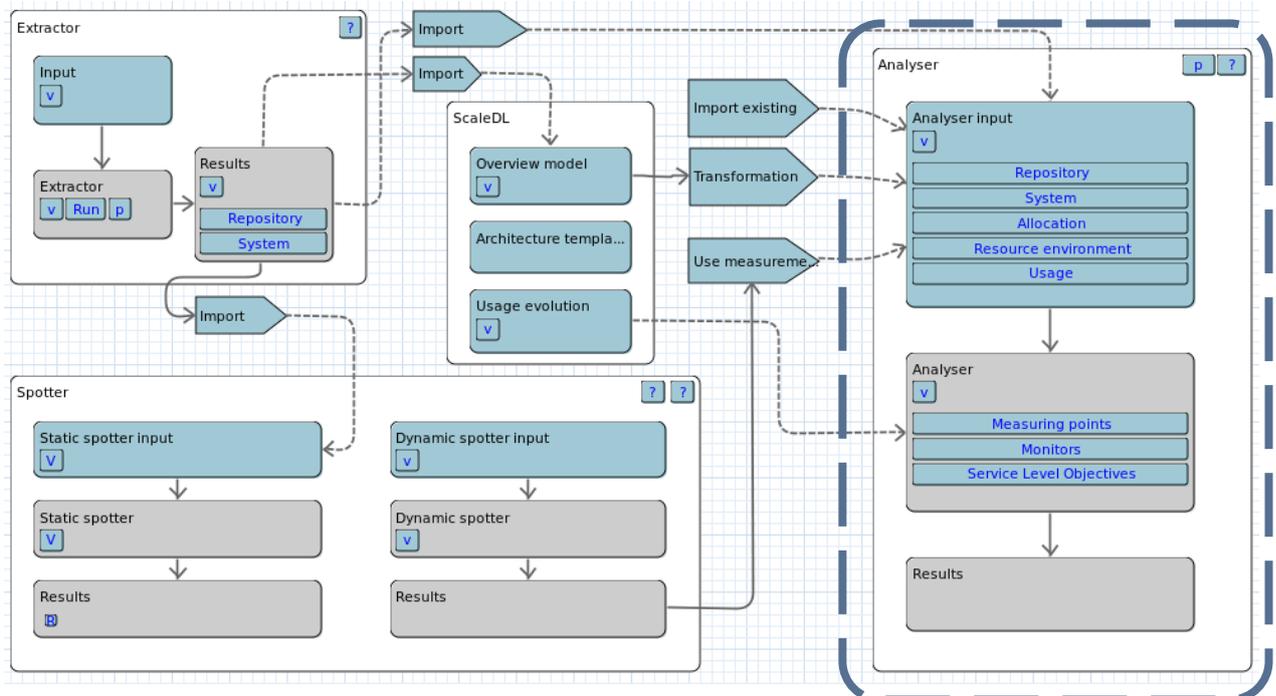


Figure 3: Analyser workflow

Problem

Scalability, elasticity, and cost-efficiency problems are often detected too late, i.e., during system testing and/or system operation. This approach leads to the risk of SLO-violating systems and expensive re-implementations. Moreover, different design variants remain uninvestigated because it would be too extensive to implement and test all potential design variants.

Features

CloudScale's Analyser measures several scalability, elasticity, and cost-efficiency metrics based on a ScaleDL instance as an input. The metrics of interest can be configured by software architects.

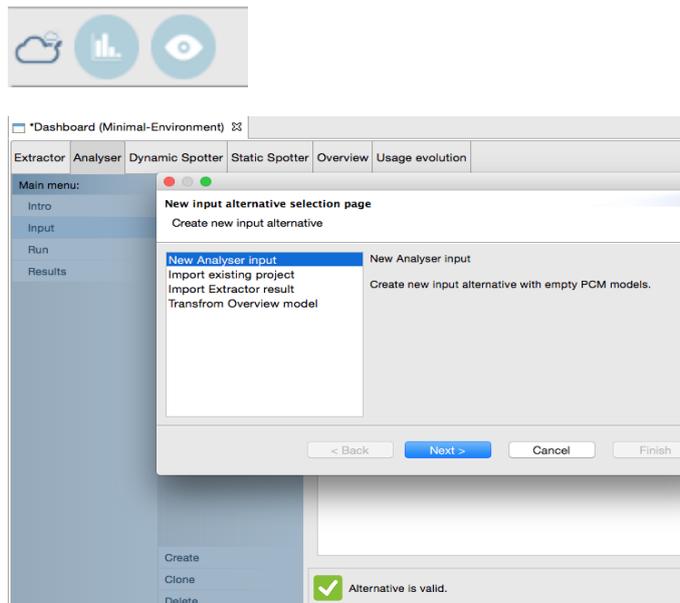
Input & Output

As an input, Analyser takes a ScaleDL instance (Extended PCM, Usage Evolution, Architectural Templates, ScaleDL Overview) and its configuration via an Experiment Automation instance. The latter is a dedicated configuration model for the Analyser, e.g., allowing to configure metric measurements and how often and how long Analyser runs are conducted.

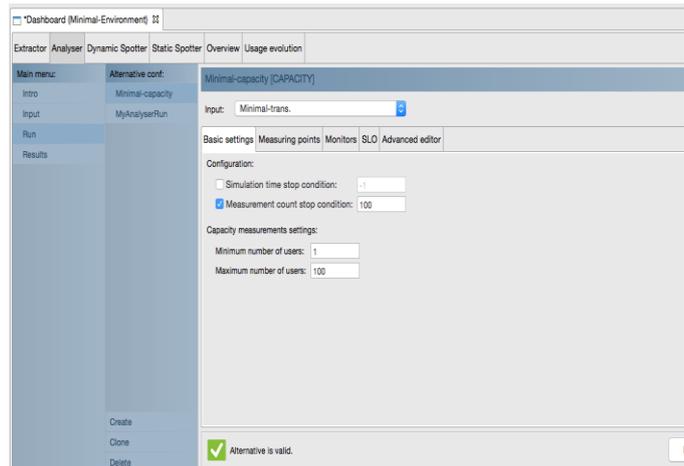
The output of the Analyser are analysis results. The CloudScale Environment can store, load, and visualize these results.

Walkthrough

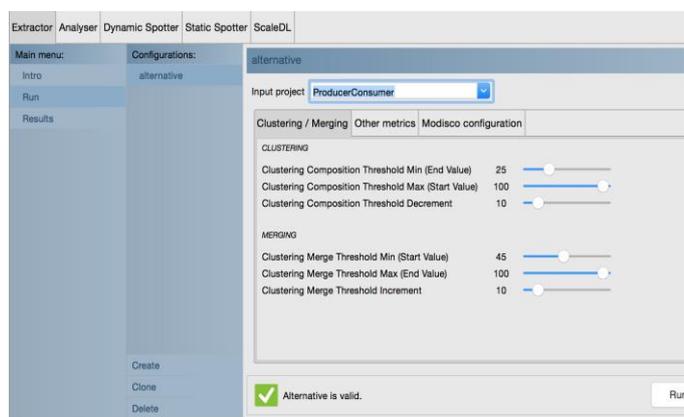
1. Switch to the CloudScale Perspective (first button of the image below).
2. Load your input ScaleDL instance into the Analyser by using the "Create" wizard in the "Input" tab. You can choose between different input sources for your ScaleDL instance as shown in the image below.



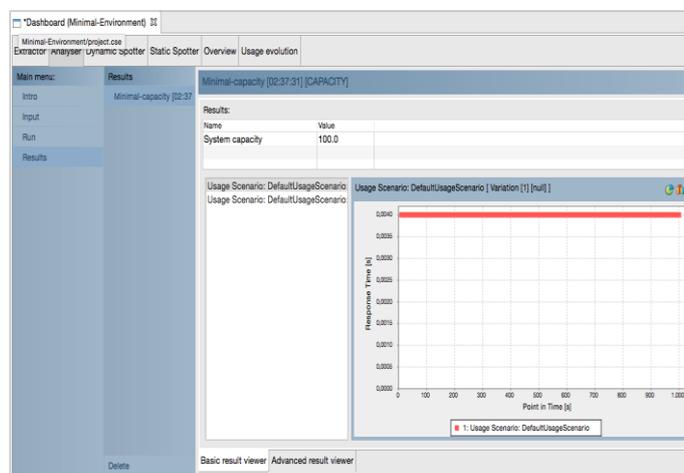
- Configure an Analyser Run via the “Run” tab and configure Analyser parameters. For example, the image below sets a stop condition for the Analyser (it stops as soon as 100 measurements have been taken).



- Run the Analyser by pressing the “Run” button and wait for its completion (the console output states “Workflow engine completed task” on completion).



- Investigate analysis results via the “Results” tab



References

The CloudScale Method includes a dedicated step for running the Analyser and for investigating its analysis results.

Static Spotter

Introduction

The Static Spotter is a reverse engineering tool based on the Reclipse for automatic detection of so called search patterns which then are interpreted as a potential scalability anti-patterns. All scalability anti-patterns are defined in the pattern catalogue. Later on, the Static Spotter is searching for the anti-patterns according to the pattern catalogue.

Relation to workflow

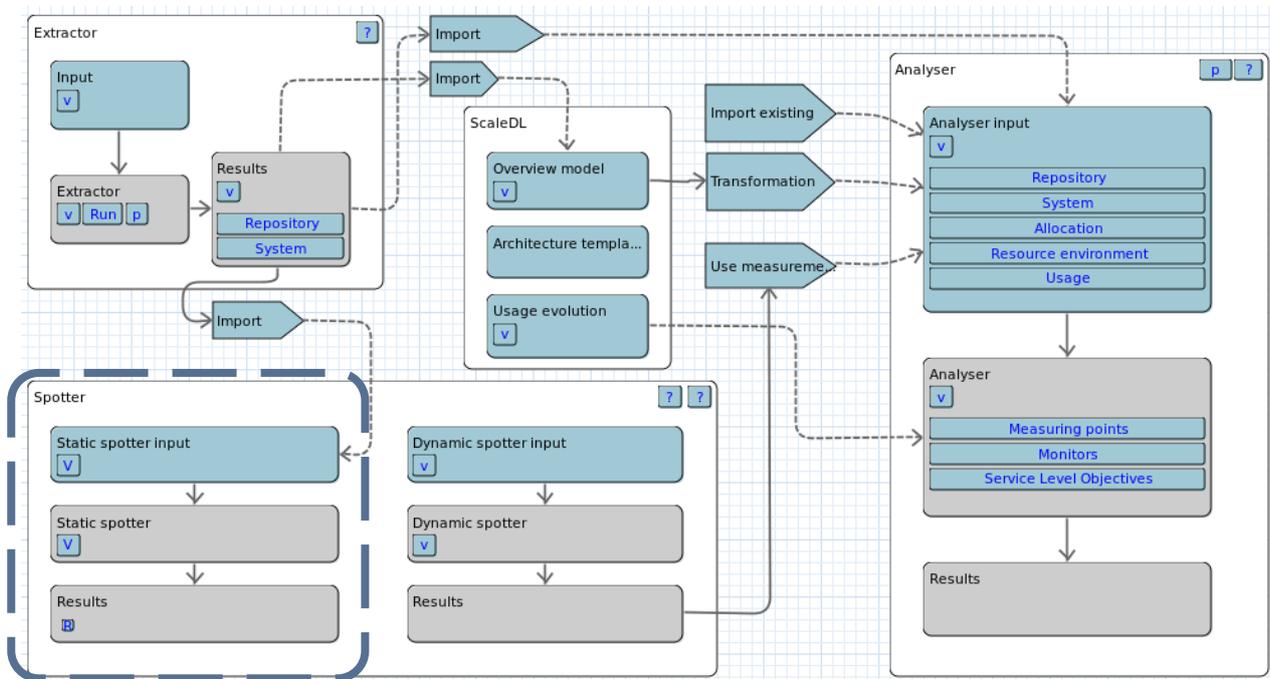


Figure 4: StaticSpotter workflow

Problem

A software system suffers scalability or performance issues while being development without consideration of scalability or performance. This would become even worse when being migrated to the Cloud computing environment. For example, the system gets and pays more resources from the Cloud providers but does not gain in its performance.

Features

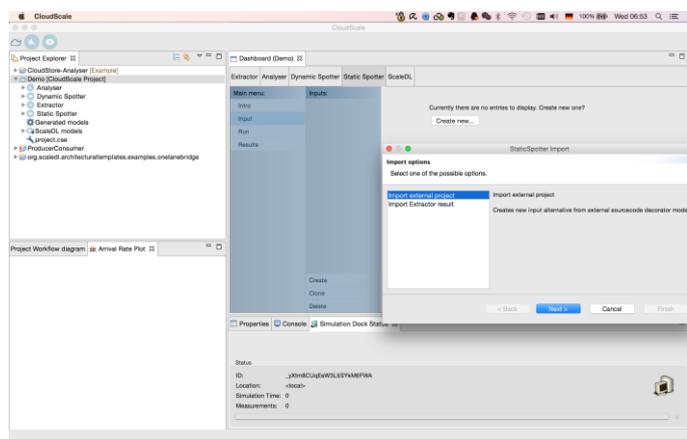
The Static Spotter exams the ScaleDL model. It uses predefined pattern or anti-pattern catalogues to detect patterns and anti-patterns automatically.

Input & Output

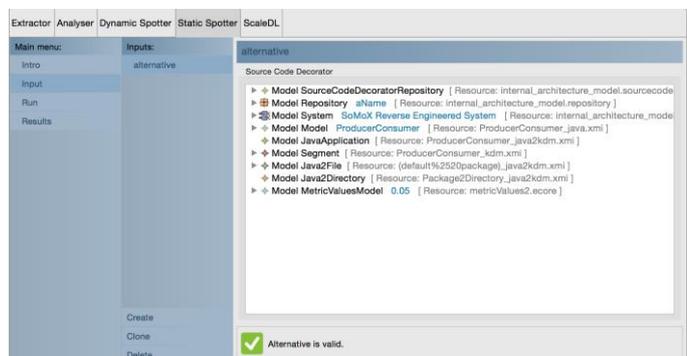
The Static Spotter takes ScaleDL models as input. After searching for predefined patterns or anti-patterns, it generates a list of pattern or anti-pattern candidates. The candidates could be methods, classes or components.

Walkthrough

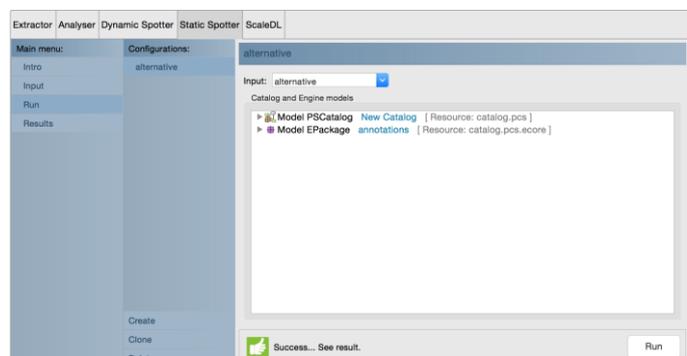
1. Go to Static Spotter Tab, click input=> create.



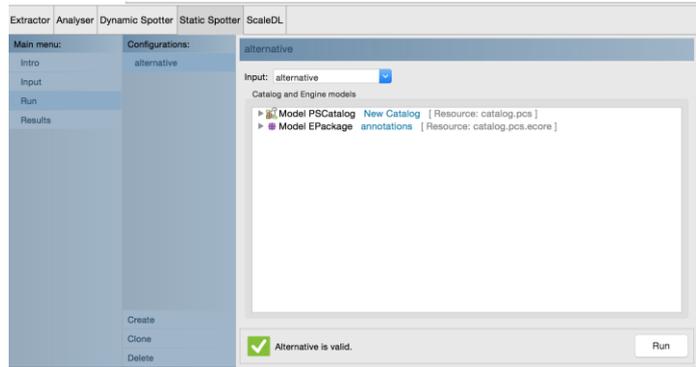
2. Select Import Extractor result



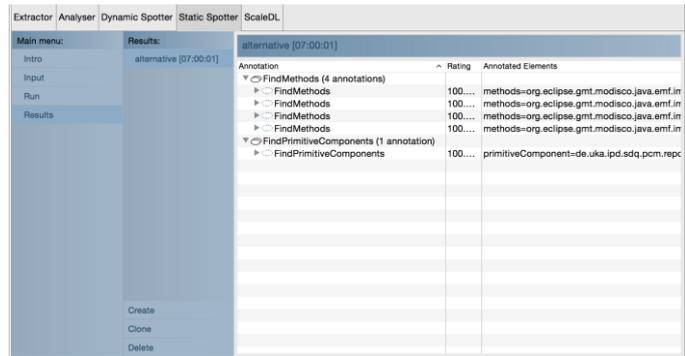
3. Click Run => Create. Select Input.



4. Run



5. Go to Results under Main menu and see the results.



Dynamic Spotter

Introduction

Dynamic Spotter is a framework for measurement-based, automatic detection of software performance problems in Java-based enterprise software systems. It combines the concepts of software performance anti-patterns with systematic experimentation.

Reference to workflow

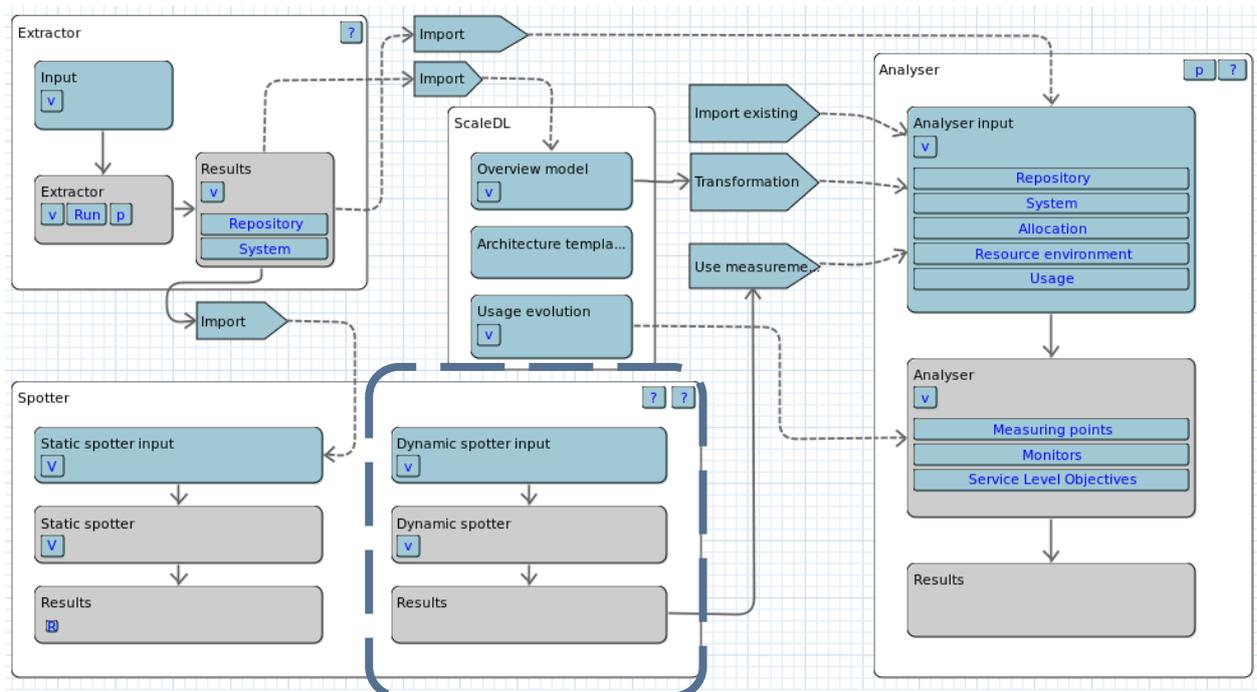


Figure 5: StaticSpotter workflow

Problem

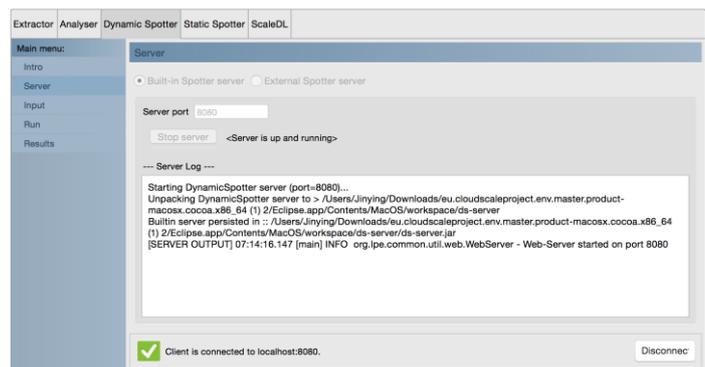
A software system suffers scalability or performance issues while being development without consideration of scalability or performance. This would become even worse when being migrated to the Cloud computing environment. Although Static Spotter is able to find potential scalability or performance anti-patterns, it is hard to tell which one is affecting actually.

Features

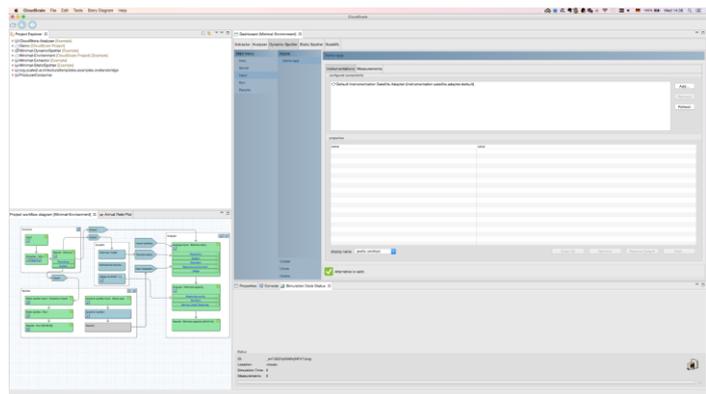
Dynamic Spotter uses systematic experimentation, analyses measurement and detects the actual affecting anti-patterns in the system.

Walkthrough

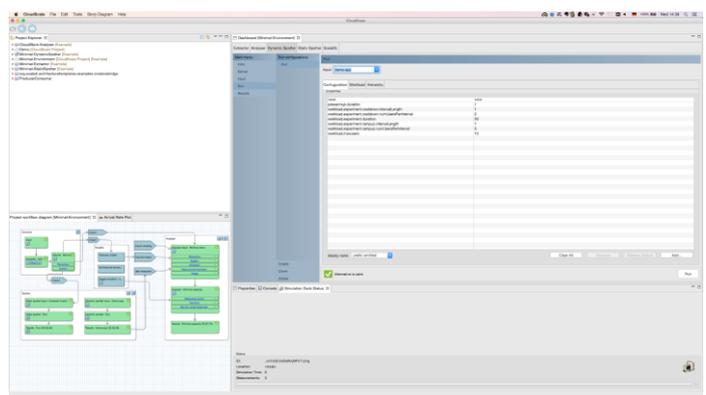
1. In order to run Dynamic Spotter, a running server is needed. Go to Dynamic Spotter Tab, Click Server under Main menu, Start Built-in Spotter server, Click Connect



2. Click Input => Create, add Instrumentation and Measurement respectively.



3. Run



4. Click Results, the result is shown as follows.

