ABSTRACT
Modern software systems are expected to deliver reliable performance under highly variable load intensities while at the same time making efficient use of dynamically allocated resources. Conventional benchmarking frameworks provide limited support for emulating such highly variable and dynamic load profiles and workload scenarios. Industrial benchmarks typically use workloads with constant or stepwise increasing load intensity, or they simply replay recorded workload traces. In this paper, we present LIMBO - an Eclipse-based tool for modeling variable load intensity profiles based on the Descartes Load Intensity Model [3] as an underlying modeling formalism.

Categories and Subject Descriptors
C.4 [Computer Systems Organization]: Performance of Systems—Modeling Techniques

General Terms
Benchmarking, Workload, Modeling

Keywords
Load Intensity Variation, Load Profile, Open Workloads, Meta-Modeling, Transformation, Model Extraction

1. INTRODUCTION
Today’s cloud and web-based IT services need to handle huge amounts of concurrent users. Customers access services independently of one another and expect reliable quality-of-service under highly variable and dynamic load intensities. In this context, any knowledge about a service’s load intensity profile is becoming a crucial information for managing the underlying IT resource landscape. Load profiles with large amounts of concurrent users are typically strongly influenced by human habits, trends, and events. This includes strong deterministic factors such as time of the day, day of the week, common working hours and planned events.

Common benchmarking frameworks such as Faban1, Rain [1], and JMeter2 allow job injection rates to be configured either to constant values, stepwise increasing rates (e.g., for stress tests), or rates based on recorded workload traces. The tool we present in this paper aims at closing the gap between highly dynamic load intensity profiles observed in real life and the current lack of support for flexible handling of variable load intensities in benchmarking frameworks.

In [3], we introduce two modeling formalisms at different abstraction levels: At the lower abstraction level, the Descartes Load Intensity Model (DLIM) offers a structured and accessible way of describing the load intensity over time by editing and combining mathematical functions. The High-Level DLIM (HLDLIM) allows the description of load variations using few defined parameters that characterize the seasonal patterns, trends, as well as bursts and noise elements.

In this demo paper, we present LIMBO3 - an Eclipse-based tool for handling and instantiating load intensity models based on DLIM. LIMBO offers an accessible way of editing DLIM instances and extracting them from existing traces. It also supports using HLDLIM parameters for easy creation of new DLIM instances through a model creation wizard. An example load profile of a DLIM instance is shown in Fig.1.

Figure 1: An example load profile of a DLIM instance plotted by LIMBO.

2. DEFINITION OF LOAD INTENSITY
In this work, load intensity is a discrete function describing arrival rates of workload units over time. We assume

1Faban http://faban.org
2JMeter http://jmeter.apache.org
3LIMBO http://www.descartes-research.net/tools/
that the work units are of a homogeneous type and define the arrival rate \( r(t) \) at time \( t \) as follows:

\[
r(t) = R'(t)
\]
where \( R(t) \) is the amount of all work units \( u_{t_0} \), with their respective arrival time \( t_0 \), that have arrived up until time \( t \).

3. LIMBO

LIMBO allows editing of load intensity models based on DLIM and supports guided model creation using the parameters defined in HLDLIM.

3.1 Descartes Load Intensity Model

DLIM describes request arrival rates over time and offers a way to define a piece-wise mathematical function for the approximation of variable arrival rates with support for (partial) periodicity, flexibility and composability.

3.2 High-Level DLIM

HLDLIM offers abstracted knowledge about load intensity variations modeled through a limited number of workload parameters. Inspired by the time series decomposition approach in BFAST [2], a HLDLIM instance describes a Seasonal and Trend part. Additionally, it features a Burst and Noise part.

3.3 Implementation

The tooling for DLIM and HLDLIM models is realized as a plug-in for the Eclipse IDE. It provides an editor for the creation and modification of model instances, as well as additional utilities for using the created models. Using DLIM’s EMF-generated code base as a basis, the following features have been implemented:

- **Model Evaluation**: Support for the DLIM function output calculation and manual refinement of model instances.
- **Modeling Process Assistance**: We are currently implementing an automated process for the creation and extraction of DLIM instances. So far, LIMBO provides a model instantiation guidance by means of a wizard.
- **Utilities**: Additional functionality is provided for existing DLIM instances. Including functionality for the generation of arrival rate series from a time-stamp series, and a tool that calculates the difference between an arrival rate trace and a model instance.

LIMBO consists of five individual plug-ins as visualized in Fig. 2:

![LIMBO architecture](image)

**Figure 2**: LIMBO architecture.

1. **DLIM Generator** The `dim.generator` plug-in contains the DLIM element interfaces and implementations, as well as their default utilities (e.g., for validation). It also contains model evaluation tools, as well as arrival rate and time-stamp series generators. It features two extension points:

- **Exporter** extension points supports custom implementations by implementing the `dim.exporter.IDlimExporter` interface. Default exporters are contained in the `dim.exporter` plug-in.
- **Extractor** extension point allows the addition of extractors for deriving a model instance from an existing trace. Extractors must implement `dim.reader.IDlimArrivalRateReader` for their trace parser and `dim.extractor.IDlimExtractor` for the model instance creator. Default extractors are contained in the `dim.extractor` plug-in.

2. **DLIM Generator Editor**

This plug-in contains the providers used by the editor, which provide display specific information, such as the display images and labels of model elements.

3. **DLIM Generator Editor**

The `dim.generator` plug-in contains all GUI elements and their utilities. It also contains implicit modeling process knowledge in its GUI.

4. **DLIM Exporter**

The `dim.exporter` plugin offers default implementations of the `dim.generator` plugin’s `dim.exporter.IDlimExporter` interface and the `exporter` extension point.

5. **DLIM Extractor**

The `dim.extractor` plug-in offers default implementations of the `dim.extractor.IDlimExtractor` interface and the `extractor` extension point. Both `extractor` extension point implementations in this plug-in use the provided default `dim.reader.ArrivalRateReader` provided by the `dim.extractor` plug-in.

4. CONCLUSIONS

In this demo paper, we introduce LIMBO: A toolkit for creating and editing of DLIM instances. By enabling the flexible handling of load intensity profiles, we address a strong need in the areas of benchmarking and elastic capacity management. Currently, we are extending LIMBO to support an automatic model-from-trace extraction process.

5. REFERENCES

