Sisyphos is done

Load-Tests with the benerator®

Volker Bergmann

volker@ databene.org
www.databene.org

Copyright © 2007 Volker Bergmann

Sisyphos is done

- **testing like Sisyphos**
- benerator to the rescue
- concepts
- populating databases
- generating files
- customizing benerator
- the road ahead

Copyright © 2007 Volker Bergmann

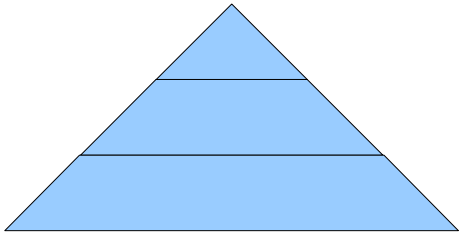
Functional Tests

- Check if the system under test behaves according to the functional requirements
- Need to be performed incrementally
- Are (usually) performed by different people
- Test on valid and invalid data

Integration Tests: QA Department

Component Tests: Testers of each team

Unit Tests: Developers



SQS

DBUnit

JUnit

Copyright © 2007 Volker Bergmann

Non-Functional Tests...

- check if the system under tests fulfills non-functional requirements, e.g. load tolerance, performance, security, etc.
- are performed after development
- Thus, leave little time for fundamental software fixes
- mostly test on valid data
- are expensive when done:
  - Experience needs to be established/bought
  - Mirror environment may be required
  - Sophisticated tools may be needed
- ...but may become extremely expensive when they are not done

Copyright © 2007 Volker Bergmann

the pain of testing

- functional and non-functional tests
  - have little in common
  - are performed with separate tools
  - base on separate data and configuration files
  - Must be maintained independently on software changes
  - Maintenance (changing) of test data files is painful: Sisyphus' rock rolls down again and again
  - data export from functional tests to load tests mostly by exporting and repeating few data sets many times
  - meaningful load tests require weeks of data definition

Copyright © 2007 Volker Bergmann

Sisyphos is done

- testing like Sisyphos
- **benerator to the rescue**
- concepts
- populating databases
- generating files
- customizing benerator
- the road ahead

Copyright © 2007 Volker Bergmann

databene benerator

- benerator
  - is an open source generator for test data
  - started in June 2006
  - is in version 0.3 (as of Oct-2007)
  - welcomes contributors
  - dual license (GPL and commercial)
- databene
  - is a placeholder for a company name
- databene and benerator are registered trademarks

Copyright © 2007 Volker Bergmann

the joy of testing

- benerator...
  - shifts data generation effort from programming/editing to configuring
  - reduces the effort of defining initial data creation from weeks to days (or hours)
  - reduces maintenance effort on software changes from hours to minutes
  - is useful for all kinds of tests, showcases and deployment setups
  - supports reuse of functional test data for load tests
  - provides performing load test *during* development, allowing fixes in earlier stages
  - enables more realistic tests (or at least load-test a system at all!)

Copyright © 2007 Volker Bergmann

Sisyphos is done

- testing like Sisyphos
- benerator to the rescue
- **concepts**
- generating files
- customizing benerator
- populating databases
- the road ahead

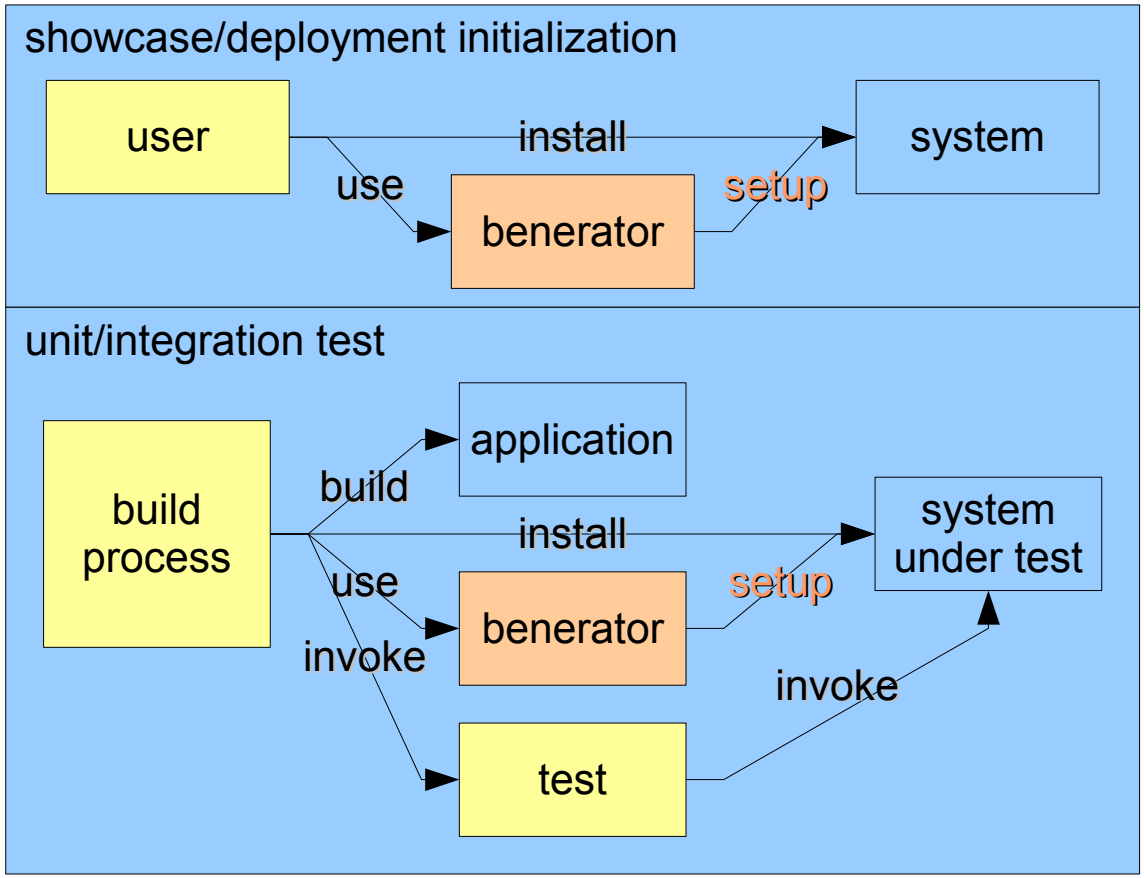
Copyright © 2007 Volker Bergmann

generation paradigm

- Benerator supports an incremental generation process:
  - predefined core data
    - provides support for unit tests
    - can be imported from systems or data files
    - may reuse data from unit tests (e.g. DBUnit setup files)
  - mass data
    - provides support for load tests
    - can be stored in systems and/or files
    - can be be created
      - efficiently
      - in arbitrary volume
      - transactionally with paging

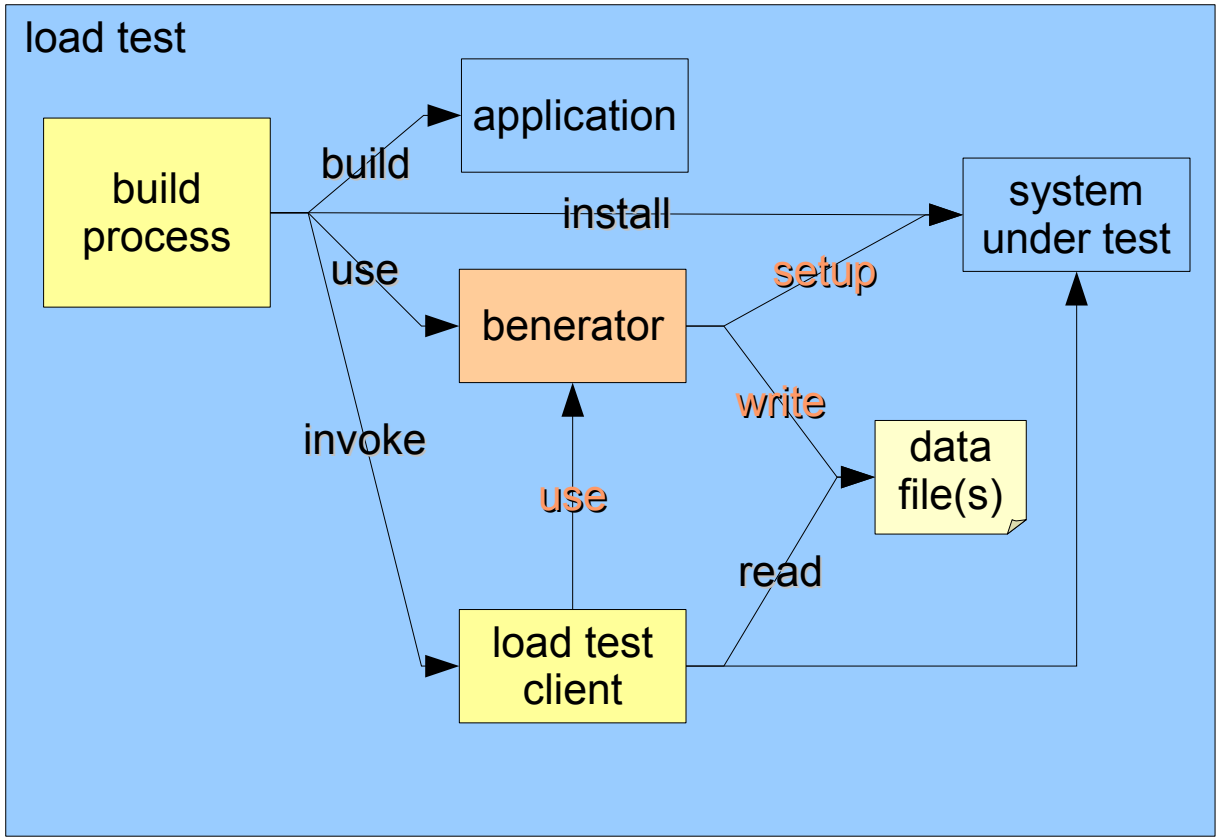
Copyright © 2007 Volker Bergmann

development process integration



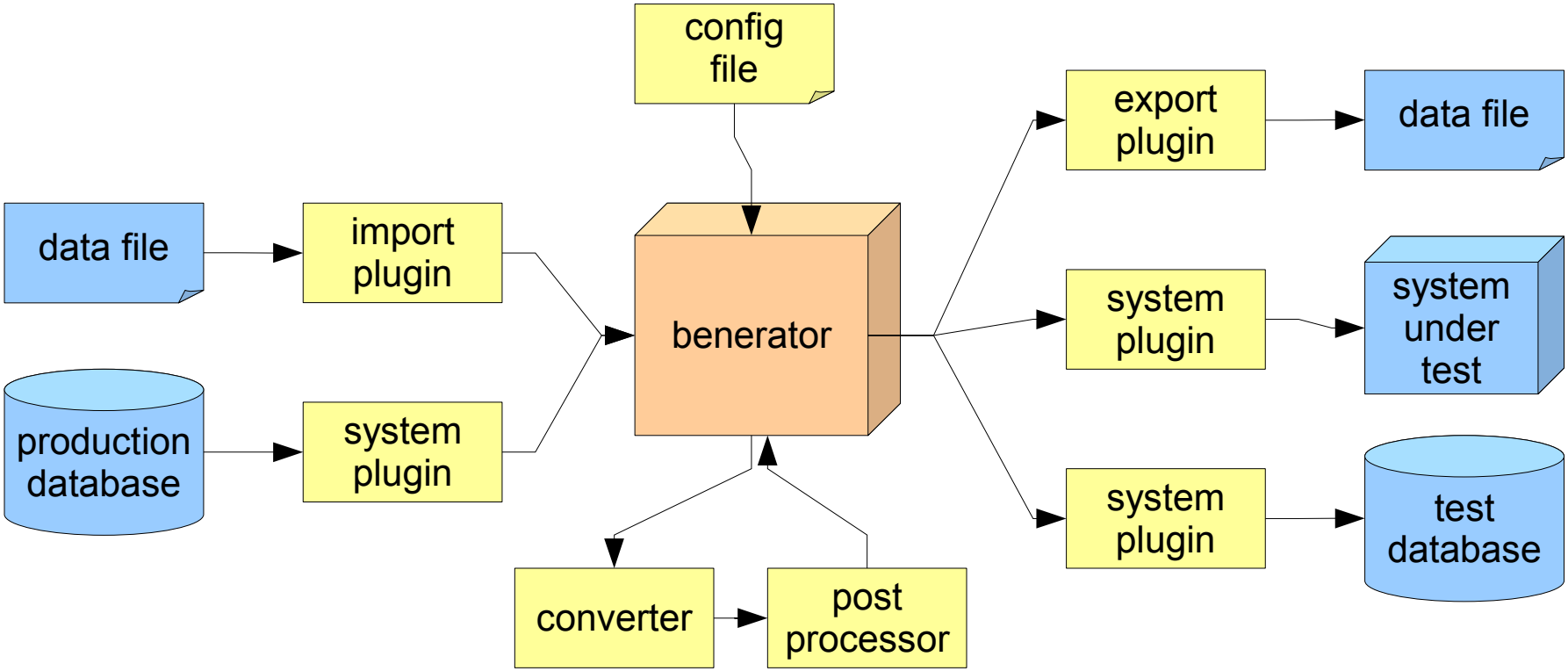
Copyright © 2007 Volker Bergmann

Load testing with benerator



Copyright © 2007 Volker Bergmann

component overview



Copyright © 2007 Volker Bergmann

features

- data generation from scratch
- importing, anonymizing and merging data from multiple sources
  - Systems: database
  - Files: CSV, flat, DBUnit
- data output to multiple targets in different formats
  - Systems: database
  - Files: SQL, CSV, flat files
- data quality assurance
  - supports single and multi-field constraints
  - ability to validate generated data with plugins

Copyright © 2007 Volker Bergmann

features

- easily extensible by plugins
- lots of predefined plugins components
- powerful randomization features (also extensible)
- processing of high volume data
- performance (at least 1 million datasets per hour on common developer hardware)
- easy to use
  - by configuration file
  - by API

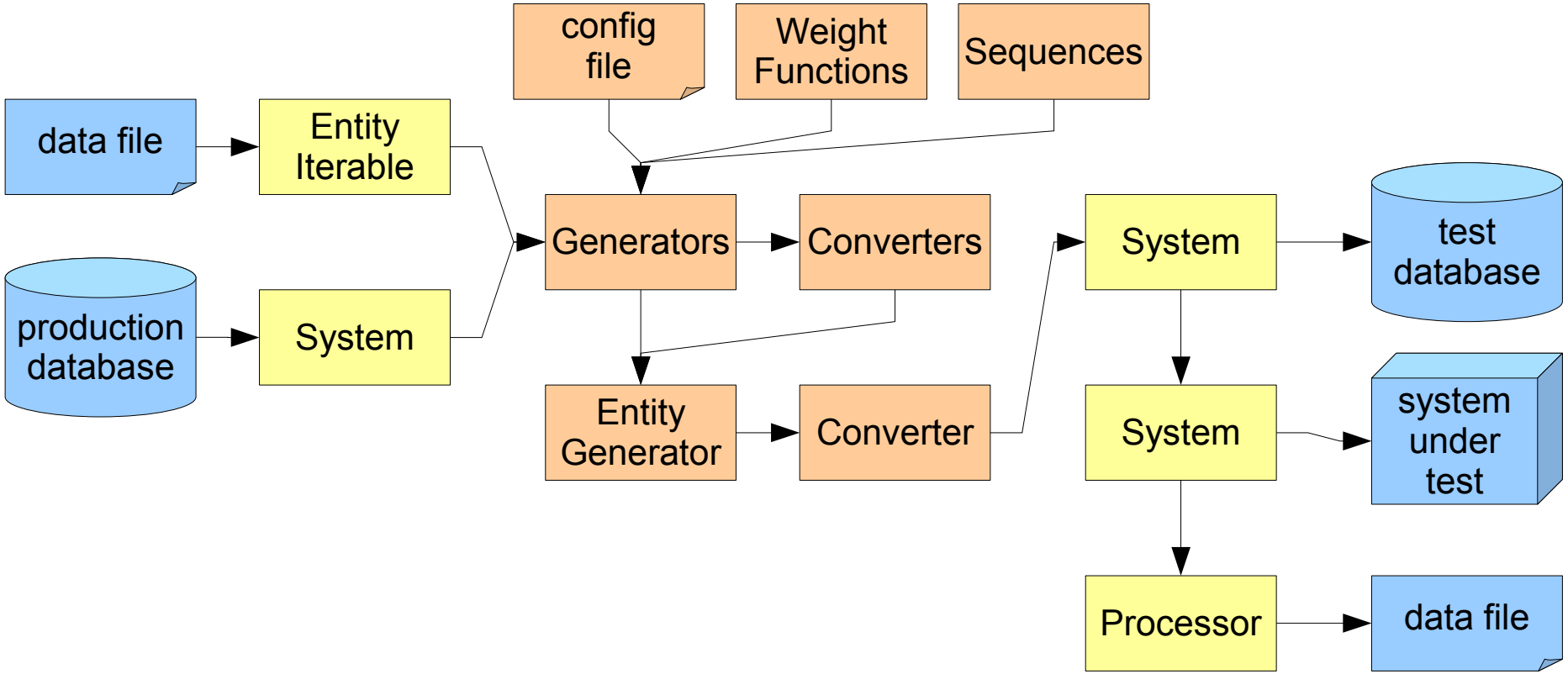
Copyright © 2007 Volker Bergmann

domain packages

- benerator provides domain packages which can create business domain data out-of-the-box
- domain packages may support different regions (e.g. countries) and locales (languages)
- the following domain packages exist (in v0.3):
  - personal: salutation, title, name, gender (D,F,I,GB,USA)
  - address: postal address (D)
- Planned packages:
  - web (v0.4)

Copyright © 2007 Volker Bergmann

component collaboration



Copyright © 2007 Volker Bergmann

wording

- An **Entity** composes several **Attributes** to a composite data unit
- **Data** can be a primitive data element (e.g. number/string) or an Entity
- A **System** is a standalone application that can store and query data and or entities (e.g. database, SAP, Siebel)

Copyright © 2007 Volker Bergmann

Sisyphos is done

- testing like Sisyphos
- benerator to the rescue
- concepts
- **populating databases**
- generating files
- customizing benerator
- the road ahead

Copyright © 2007 Volker Bergmann

using JavaBeans

- Benerator uses a spring-like syntax for instantiation JavaBeans
- JavaBeans are shared by a Context object
- benerator imports meta data from systems, e.g. databases

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE setup SYSTEM "http://databene.org/benerator-0.3.dtd">
<setup>

    <bean id="db" class="org.databene.platform.db.adapter.DBSystem">
        <property name="url" value="jdbc:oracle:thin:@localhost:1521:XE"/>
        <property name="driver" value="oracle.jdbc.driver.OracleDriver"/>
        <property name="user" value="benerator"/>
        <property name="password" value="benerator"/>
    </bean>
```


Copyright © 2007 Volker Bergmann

sql scripts and data import

- SQL scripts can be excuted on a database:

```
<run-task class="org.databene.platform.db.adapter.RunSqlScriptTask">
    <property name="uri" value="shop/create_tables.oracle.sql"/>
    <property name="db" ref="db"/>
</run-task>
```

- Entities can be imported from data files (CSV, Flat, DBUnit):

```
<create-entities source="shop/shop.dbunit.xml">
    <processor ref="db"/>
</create-entities>
```


Copyright © 2007 Volker Bergmann

creating data

- benerator imports database metadata (table structures)
- From the metadata, attribute generators are created that fulfill the database constraints

```
<create-entities name="db_product" count="1000" pagesize="500">
    <processor ref="db"/>
</create-entities>
```

- The `processor ref` refers a previously defined JavaBean and makes benerator forward all generated entites to it
- If no `count` is specified, generation continues as long as all internal (attribute) generators can provide new data

Copyright © 2007 Volker Bergmann

Multiple exports

- Multiple processors and processor-refs may be used for exporting the entity to different systems, files and formats

```
<create-entities name="db_product" count="1000" pagesize="500">
    <processor ref="db"/>
    <processor class="org.databene.platform.flat.FlatFileEntityExporter">
        <property name="uri" value="products.flat"/>
        <property name="properties" value="ean_code[13],price[10.2r0]"/>
    </processor>
</create-entities>
```


Copyright © 2007 Volker Bergmann

customizing generation

- Created data fulfills all metadata constraints, the samples above generate products with EAN codes like these:

```
LWYS                GTX                 NJZJJASLEO
```

- For fulfilling application constraints, attribute generators must be configured:

```
<create-entities name="db_product" count="1000" pagesize="500">
    <attribute name="ean_code"
               generator="org.databene.domain.product.EANGenerator"/>
    <processor ref="db"/>
</create-entities>
```

- Resulting in these EAN codes:

```
26177838            42826518            6118314424819
```


Copyright © 2007 Volker Bergmann

Generating attributes

- Attributes can be generated
  - using regular expressions:

```
<attribute name="password" pattern="[A-Za-z0-9]{8,12}"/>
```

  - querying systems:

```
<attribute name="id" source="db"
    selector="SELECT seq_db_id_generator.NEXTVAL FROM dual"
    distribution="random"/>
```

  - specifying metadata

```
<attribute name="category" values="A,B,C"/>
```

```
<attribute name="price"
    min="0.49" max="999.99" precision="0.10" distribution="cumulated"/>
```

  - Lots(!) of metadata attributes are available,
    see http://databene.org/databene-benerator/file_format.html

Copyright © 2007 Volker Bergmann

multi-field-constraints

- If multi-field-constraints apply, a generator must be provided that creates objects with valid attribute combinations
- The created objects may be JavaBeans, Java Maps or Entities
- The setup file declares a generator as `variable`
- Generated objects are shared by and referenced from the benerator Context:

```
<create-entities name="db_customer" pagesize="1000">
    <variable name="person" region="DE"
              generator="org.databene.domain.person.PersonGenerator"/>
    <attribute name="salutation" source="person.salutation"/>
    <attribute name="first_name" source="person.givenName"/>
    <attribute name="last_name" source="person.familyName"/>
    <processor ref="db"/>
</create-entities>
```


Copyright © 2007 Volker Bergmann

Sisyphos is done

- testing like Sisyphos
- benerator to the rescue
- concepts
- populating databases
- **generating files**
- customizing benerator
- the road ahead

Copyright © 2007 Volker Bergmann

Hello World

Special Processor implementation support data export to CSV or flat files or FreeMarker templates:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE setup SYSTEM "http://databene.org/benerator-0.3.dtd">
<setup>
    <create-entities name="greeting" count="4">
        <attribute name="salutation" values="Hello,Hi,Howdy"/>
        <attribute name="name" values="Alice,Bob,Charly"/>
        <processor id="csv" class="org.databene.platform.csv.CSVEntityExporter">
            <property name="uri" value="names.csv"/>
            <property name="properties" value="salutation,name"/>
        </processor>
    </create-entities>
</setup>
```


Copyright © 2007 Volker Bergmann

Hello World

- Running benerator with this configuration, it creates a random output file 'names.csv' like this:

```
salutation,name
Hello,Charly
Howdy,Alice
Hello,Bob
Hi,Bob
```


Copyright © 2007 Volker Bergmann

Sisyphos is done

- testing like Sisyphos
- benerator to the rescue
- concepts
- populating databases
- generating files
- **customizing benerator**
- the road ahead

Copyright © 2007 Volker Bergmann

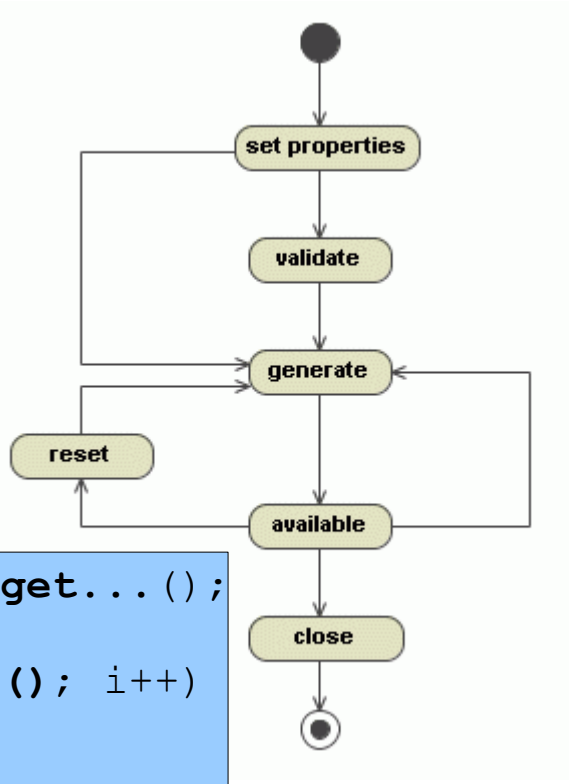
plugin contracts

- Generator — data / entity creation
- Task — arbitrary functionality
- Converter — data / entity conversion
- Sequence — number series / random functions
- WeightFunction — probability distributions
- EntityIterable — entity import
- Processor — data processing / storage
- DescriptorProvider — maps platform dependent constraints
- System — stores entities and supports queries

Copyright © 2007 Volker Bergmann

Generator contract


```
public interface Generator<E> {
    void validate();
    E generate();
    void reset();
    void close();
    boolean available();
    Class<E> getGeneratedType();
}
```




```
PersonGenerator generator = GeneratorFactory.get...();
generator.validate()
for (int i = 0; i < 10 && generator.available(); i++)
    System.out.println(generator.generate());
generator.close();
```


Copyright © 2007 Volker Bergmann

Task contract

- A Task name is used for identification.
- Life cycle is imposed by the init() and destroy() methods
- The run() method
  - implements the core functionality of the Task class.
  - may be executed sereval times in a row.

```
public interface Task
        extends Runnable {
    String getTaskName();
    void init(TaskContext context);
    void run();
    void destroy();
}
```


Copyright © 2007 Volker Bergmann

Task contract

- By default, Tasks are executed single-threaded
- If multithreaded execution is desired, the task class must implement one of the interfaces:
  - `ThreadSafe`: Task is safe for multithreaded execution
  - `Parallelizable`: Task is not thread-safe but cloned and executed in several instances in parallel
- multi-threaded execution supports paging:
  - Page start: init() is called on all task object instances
  - Page end: the same for the destroy() method
- If you need 'global' paging actions, make the Task thread-safe or use a PageListener (see customizing guide).

Copyright © 2007 Volker Bergmann

Converter contract

- A `Converter` converts one or more source type(s) to one target type
- benerator provides lots of Converter implementations
- The `AnyConverter` can be usedfor conversion to arbitrary types

```
public interface Converter<S, T> {
    Class<T> getTargetType();
    T convert(S sourceValue);
}
```


Copyright © 2007 Volker Bergmann

Sequence contract

- A Sequence is used for providing random numbers or deterministic number series
- Benerator provides some implementations: random, shuffle, cumulated, randomWalk, step
- A custom sequence is defined by providing an implementation of
  - `AbstractDoubleGenerator`
  - and (optionally) `AbstractLongGenerator`
- The generator implementations may have up to two parameters, they must be properties named `variation1` and `variation2`

Copyright © 2007 Volker Bergmann

Sequence contract

- The sequence can be instantiated programmatically:

```
new Sequence("mySequence", new my.MyDoubleGenerator())
```

- ...or in a file `org/databene/benerator/sequence.properties`:

```
mySequence=my.MyDoubleGenerator
```


Copyright © 2007 Volker Bergmann

WeightFunction contract

- Weight functions may be used for generating numbers or specifying probability distribution of sample values:

```
public interface WeightFunction extends Distribution {
    double value(double param);
}
```

- benerator normalizes the function over the specified interval (min-max) and resolution
- A weight function may have up to two parameters, they must be properties named `variation1` and `variation2`

Copyright © 2007 Volker Bergmann

Sisyphos is done

- testing like Sisyphos
- benerator to the rescue
- concepts
- generating files
- customizing benerator
- populating databases
- **the road ahead**

Copyright © 2007 Volker Bergmann

limitations of v0.3

- Proof-of-concept version
- Java 5.0
- Just primitive SQL types
- Tested with
  - Oracle 10g (thin driver)
  - MySQL 5
- Incomplete/not final:
  - File format
  - uniqueness/identity concept, key generation
  - SQL types
  - region concept

Copyright © 2007 Volker Bergmann

next steps

- Q4/2007: business development
  - establishing support services
  - feedback evaluation
- Q1/2008: Release 0.4
- Q3/2008: Release 1.0

Copyright © 2007 Volker Bergmann

planned/possible features

- Platform support: JavaBean, Hibernate, XML
- Domains: web, address (USA)
- Finalizations: SQL types, key generation, identity, uniqueness, region concept
- Assuring support for further JDBC drivers
- Improved support for partial data generation (for unit tests)
- Developer plugins: ant/maven, Eclipse
- Further system adapters, e.g. Siebel, SAP

Copyright © 2007 Volker Bergmann

how you can contribute

- evaluate benerator
- give feedback
  - request features
  - report bugs
- use generator in your project
  - contribute fixes or enhancements
  - contribute your custom domain packages
- buy
  - support services
  - commercial licenses

Copyright © 2007 Volker Bergmann

...questions?

Thank you!

Volker Bergmann
volker@ databene.org
http://databene.org

Copyright © 2007 Volker Bergmann