

matsimJ

An Overview of the new MATSim Implementation in Java

Marcel Rieser · VSP, TU Berlin

rieser@vsp.tu-berlin.de

2.10.2006

What we will talk about...

Overview MATSim-T processes

MATSim-T

- existing parts
- using, running iterations
- missing parts, parts open for improvement

Modules

- existing
- using
- missing

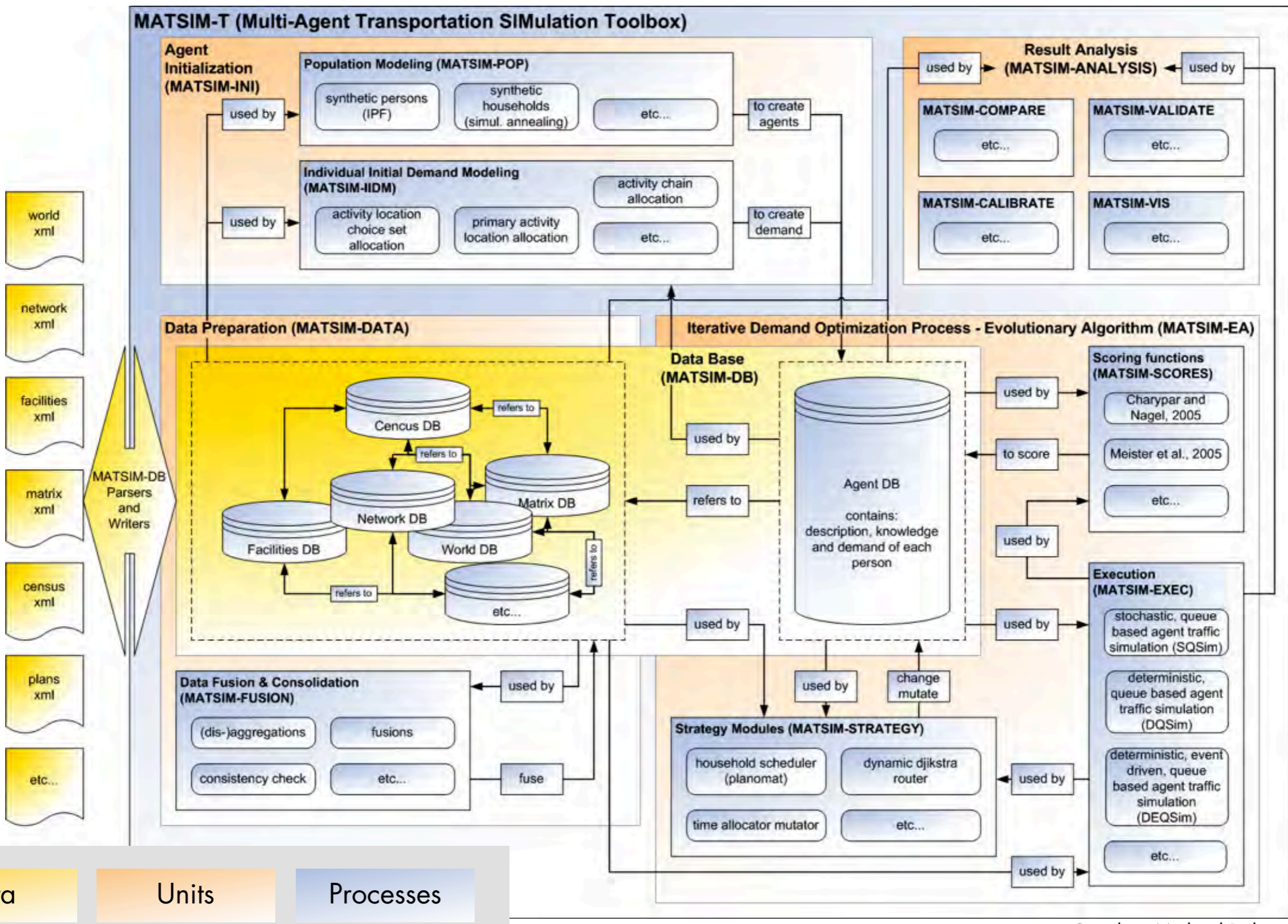
Open questions

Policy decisions, coding conventions

Proposals for workshops this week

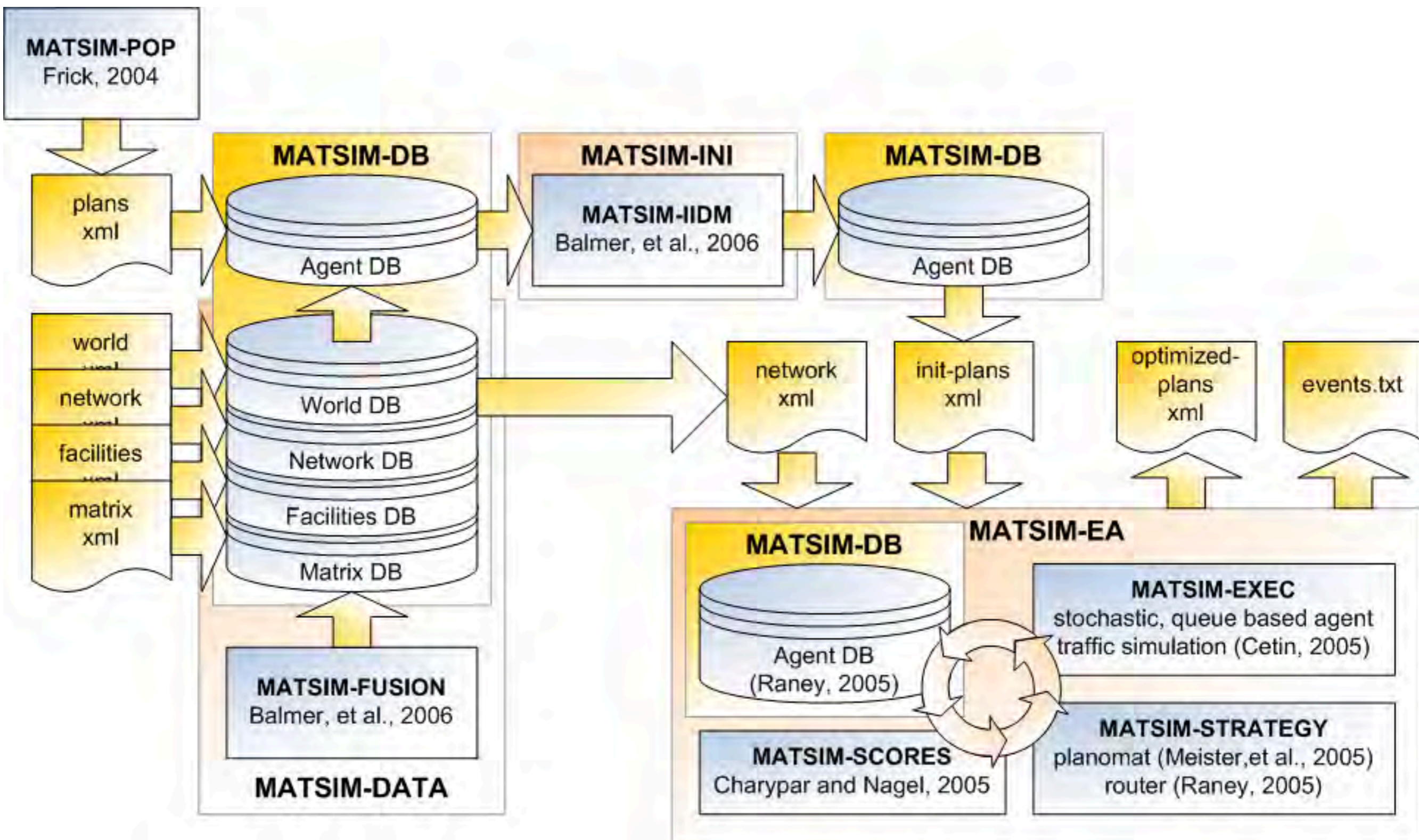
Overview

MATSim-T (detailed) Overview



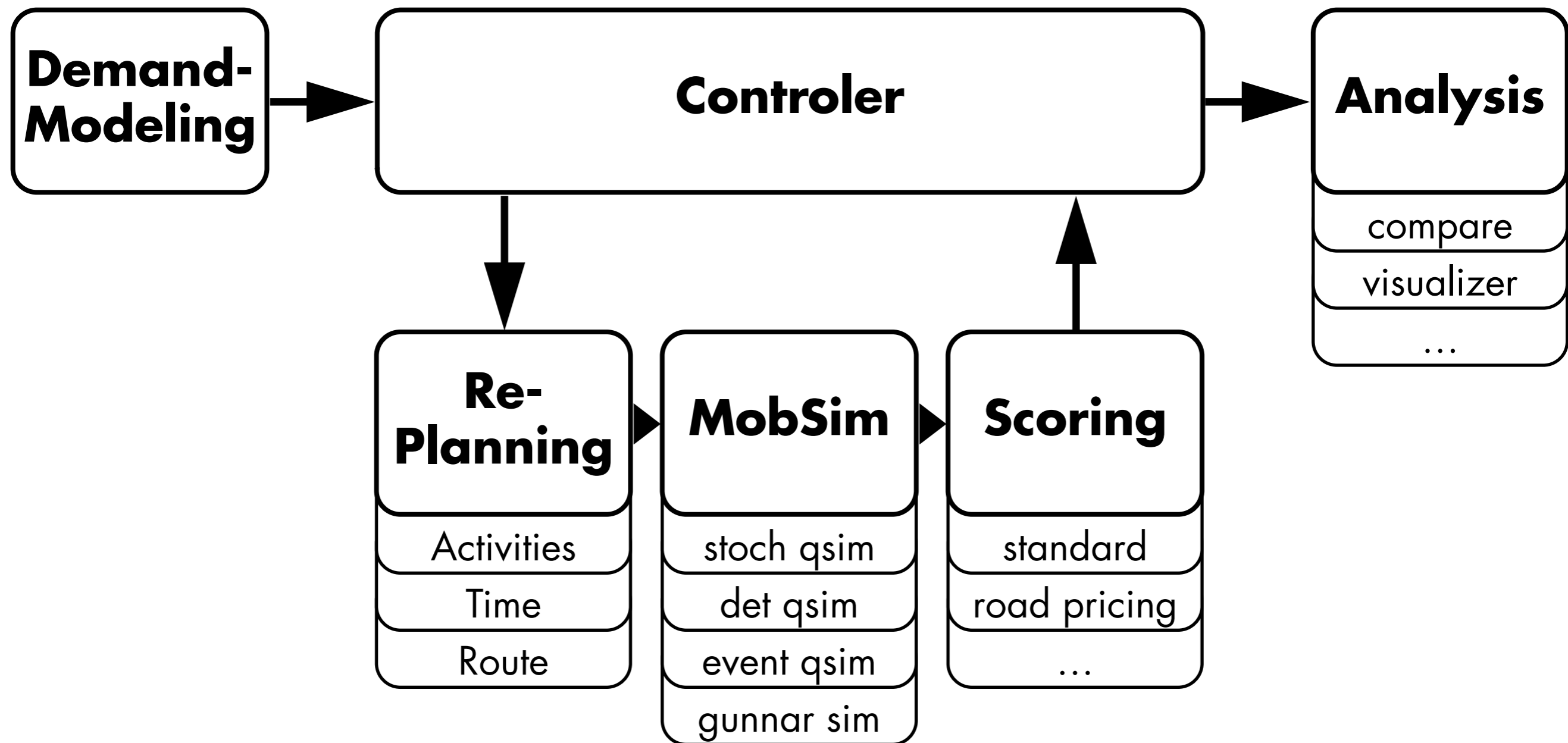
Graphic: Michael Balmer

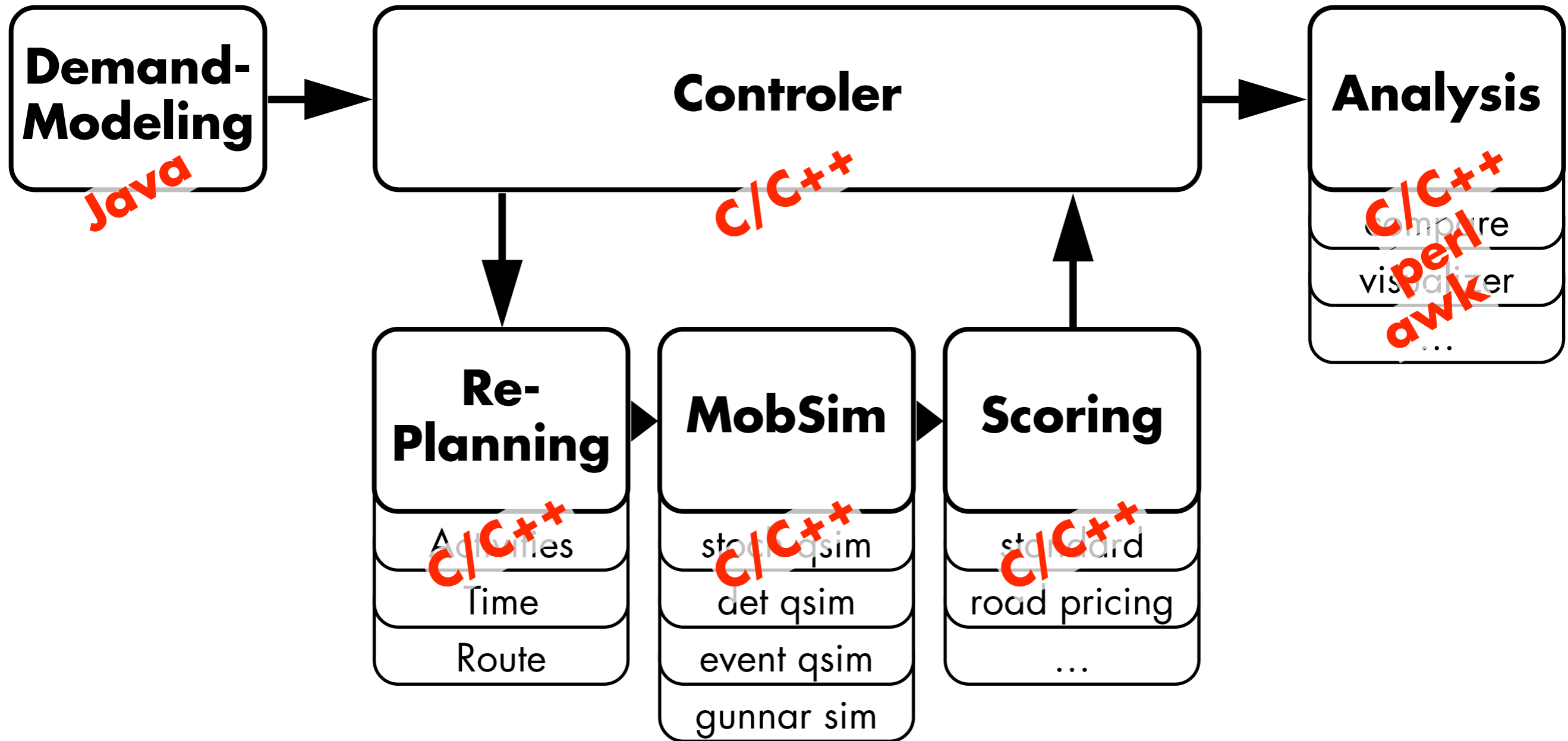


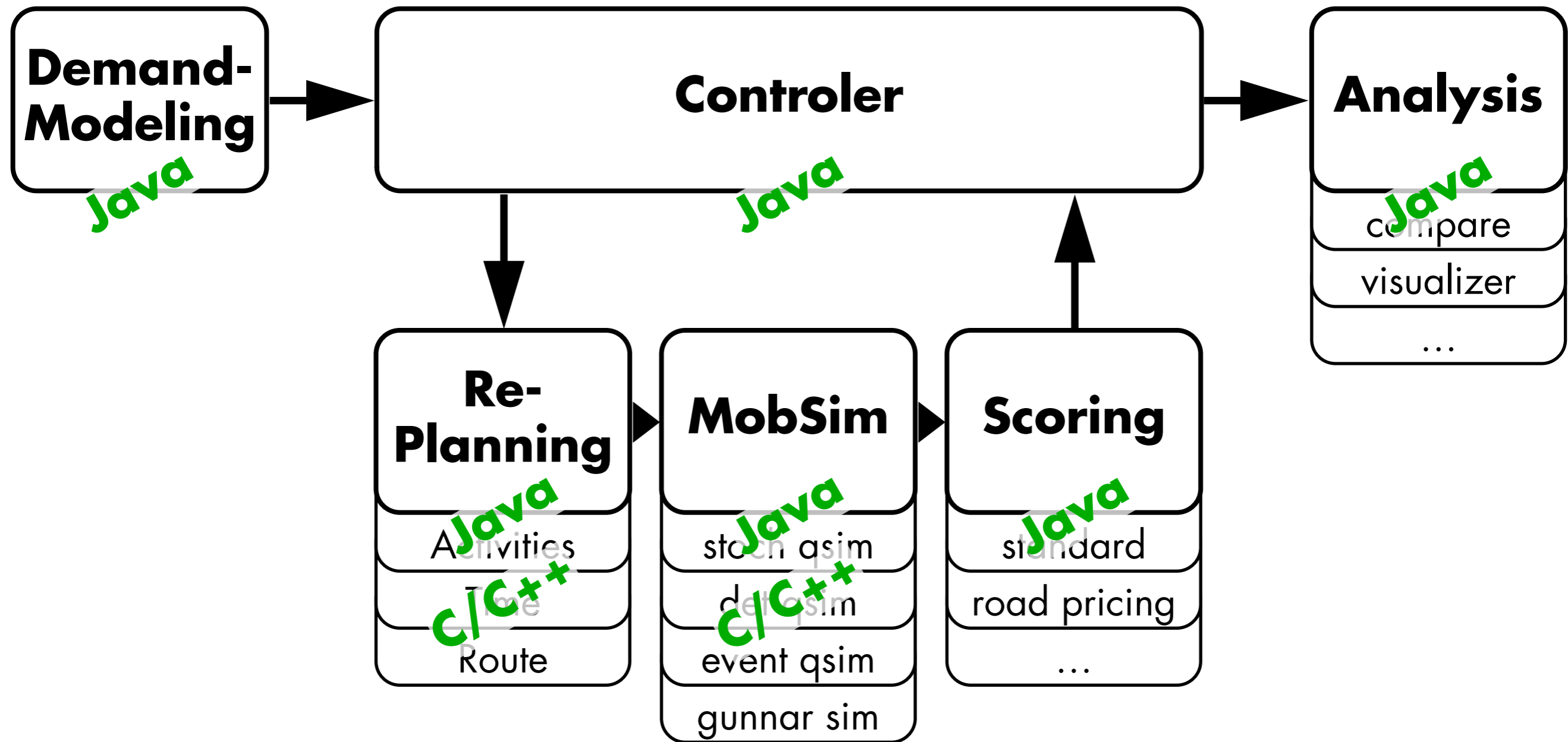


Graphic: Michael Balmer









Controler

Controler...

...runs iterations

- can be configured in config.xml

...runs mobility simulation

- currently hard-coded, but can be replaced with Java-module or external module
- two different mobsims available today: det qsim, gunnar sim
- more mobsims coming this week? e.g. event qsim?

...scores plans

- scoring algorithm currently hard-coded, but can easily be replaced with other algorithms implemented in Java
- currently one default implementation available, extended road-pricing version is in development

...organizes re-planning

- strategies can be configured in config.xml
- re-planning percentage fixed per run (maybe this could change this week?)
- currently available modules: planomat (time, activities), time allocation mutator (time), router (route), select previous plan (activities, time, route)

```
package org.matsim.demandmodeling.controller;

public class Controller {

    public Controller() {
        super();
    }

    public final void run(String[] args) {
        Gbl.createConfig(args);
        Gbl.createWorld();
        Gbl.createFacilities();

        startup();
        doIterations();
        shutdown();
    }

    protected void runMobSim() { ... }
    private final void doIterations() { ... }
    private final void startup() { ... }
    protected final void shutdown() { ... }

    public static void main(String[] args) {
        final Controller controller = new Controller();
        controller.run(args);
    }
}
```

```
java -cp MATSim_test.jar org.matsim.demandmodeling.controller.Controller config.xml
```

Config.xml

```

<?xml version="1.0" ?>
<config>
  <module name="global">
    <param name="randomSeed" value="4711" />
    <param name="outputTimeFormat" value="HH:mm" />
  </module>

  <module name="network">
    <param name="inputNetworkFile" value="input/network.xml" />
    <param name="inputVersion" value="v1" />
    <param name="outputNetworkFile" value="output/out_net.xml" />
    <param name="outputVersion" value="v1" />
  </module>

  <module name="plans">
    <param name="inputPlansFile" value="input/plans.xml" />
    <param name="inputVersion" value="matsimXMLv4" />
    <param name="outputPlansFile" value="output/plans.xml" />
    <param name="outputVersion" value="v4" />
  </module>

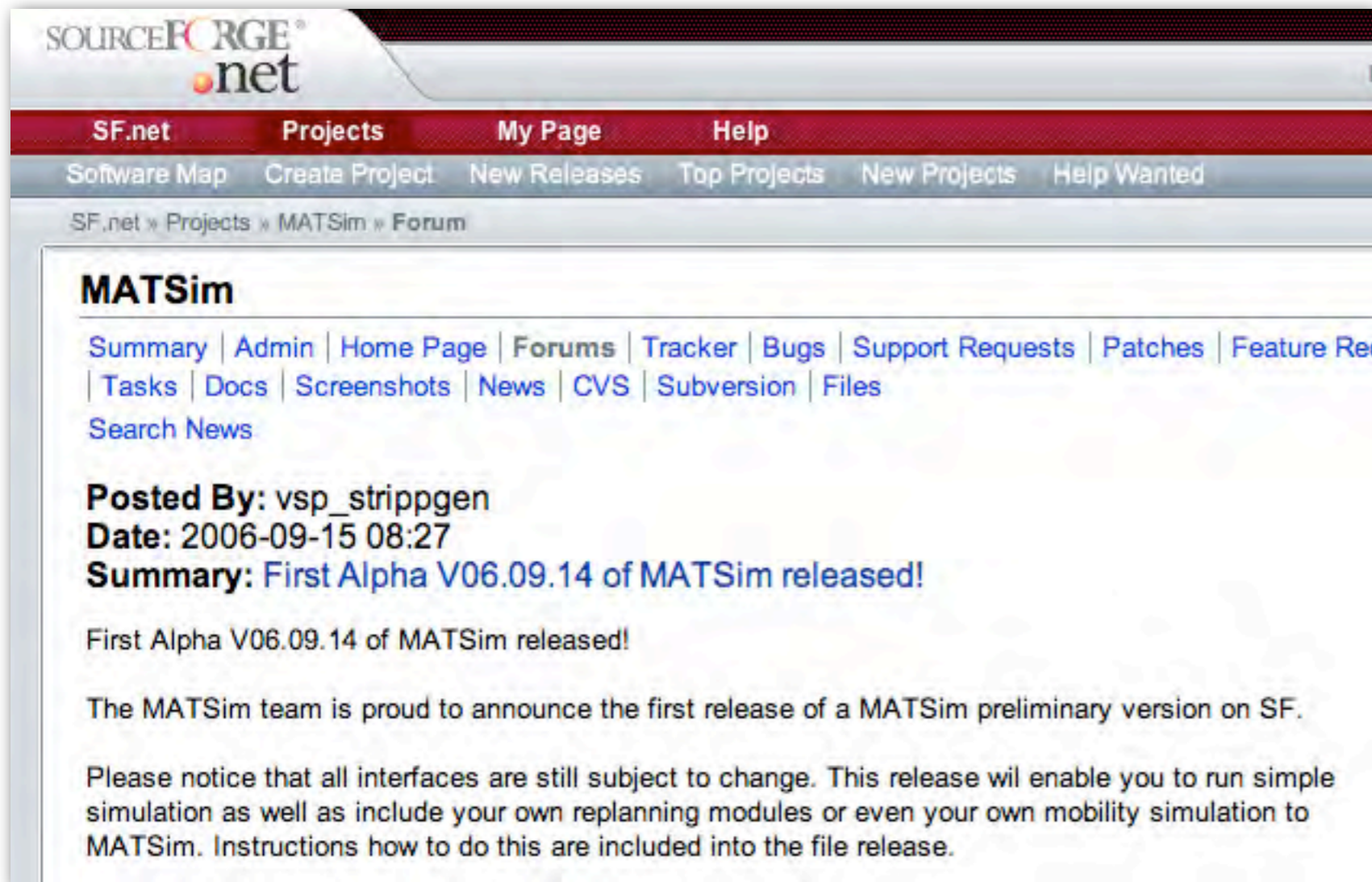
  <module name="events">
    <param name="outputFormat" value="TXT_v1" />
    <param name="outputFile" value="&CVSR00T;/&OUTPUTBASE;/out.events.txt" />
  </module>

  <module name="controler">
    <param name="outputDirectory" value="output/current" />
    <param name="lastIteration" value="1000" />
  </module>

  <module name="simulation">
    <!-- ... -->
  </module>

</config>

```



The screenshot shows a forum post on SourceForge for the MATSim project. The post is titled "First Alpha V06.09.14 of MATSim released!" and was posted by vsp_strippgen on 2006-09-15 at 08:27. The post content includes a summary and a detailed announcement from the MATSim team.

MATSim

[Summary](#) | [Admin](#) | [Home Page](#) | [Forums](#) | [Tracker](#) | [Bugs](#) | [Support Requests](#) | [Patches](#) | [Feature Requests](#) | [Tasks](#) | [Docs](#) | [Screenshots](#) | [News](#) | [CVS](#) | [Subversion](#) | [Files](#)

[Search News](#)

Posted By: vsp_strippgen
Date: 2006-09-15 08:27
Summary: [First Alpha V06.09.14 of MATSim released!](#)

First Alpha V06.09.14 of MATSim released!

The MATSim team is proud to announce the first release of a MATSim preliminary version on SF.

Please notice that all interfaces are still subject to change. This release will enable you to run simple simulation as well as include your own replanning modules or even your own mobility simulation to MATSim. Instructions how to do this are included into the file release.

(maybe we should talk about a numbering system for releases or nomenclature in general...)

The download contains:

matsim.jar

tutorial.pdf

equil-net (example network)

config.xml

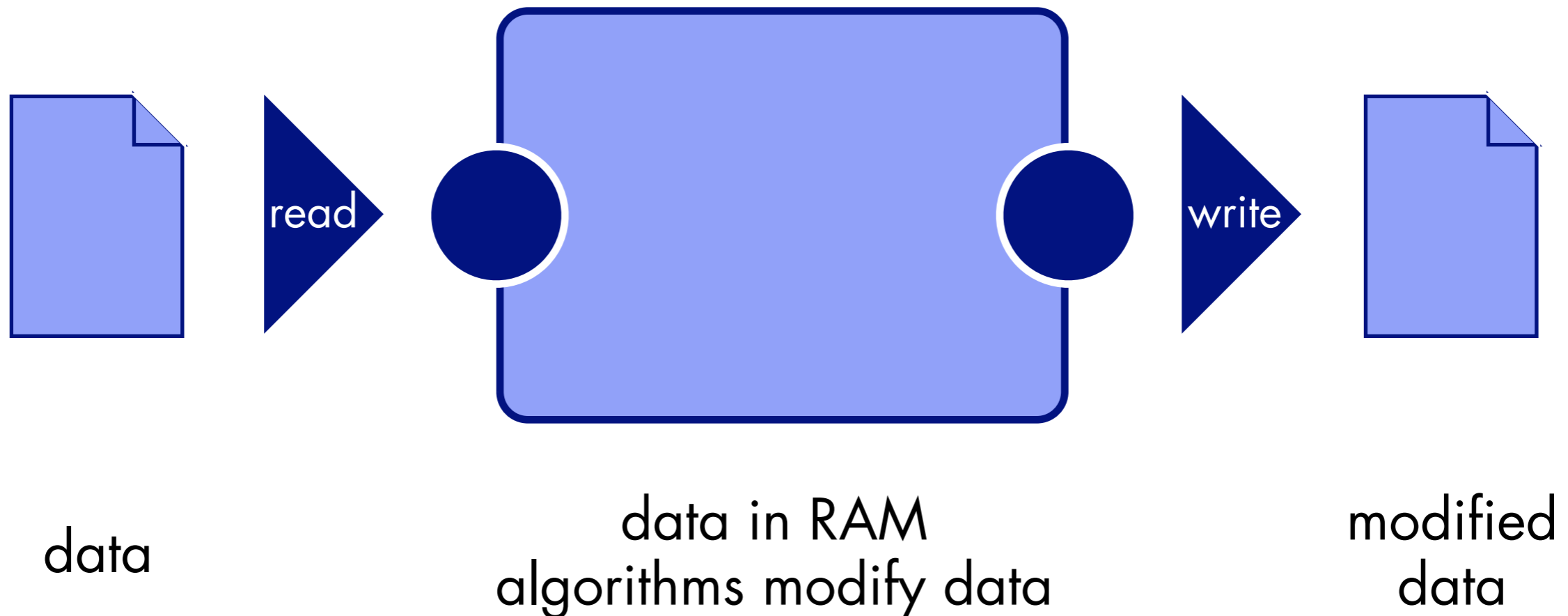
config_external_replanning.xml

config_external_mobsim.xml

Algorithms

Algorithms work on data structures

Data structures are provided, together with readers and writers



```
////////////////////////////////////  
// xy2links  
////////////////////////////////////  
  
public static void xy2links(String[] args) {  
    System.out.println("RUN: xy2links");  
    Gbl.createConfig(args);  
  
    System.out.println("  reading the network...");  
    NetworkLayer network = null;  
    NetworkLayerBuilder.setNetworkLayerType  
        (NetworkLayerBuilder.NETWORK_DEFAULT);  
    network = (NetworkLayer)World.getSingleton().createLayer  
        (NetworkLayer.LAYER_TYPE,"false",null);  
    NetworkParser network_parser = new NetworkParser(network);  
    network_parser.parse();  
    System.out.println("  done.");  
  
    System.out.println("  setting up plans objects...");  
    Plans plans = new Plans(Plans.USE_STREAMING);  
    PlansWriter plansWriter = new PlansWriter(plans);  
    plans.setPlansWriter(plansWriter);  
    PlansReaderI plansReader = PlansReaderBuilder.getPlansReader(plans);  
    System.out.println("  done.");  
  
    System.out.println("  adding plans algorithm... ");  
    plans.addAlgorithm(new XY2Links(network));  
    System.out.println("  done.");  
  
    System.out.println("  reading, processing, writing plans...");  
    plansReader.read();  
    plans.runAlgorithms();  
    plansWriter.write();  
    System.out.println("  done.");  
  
    System.out.println("RUN: xy2links finished.");  
    System.out.println();  
}
```

load configuration

read network

**setup data structure
setup reader
setup writer**

setup algorithms

**read
run algorithms
write**

```
////////////////////////////////////
// xy2links
////////////////////////////////////

public static void xy2links(String[] args) {
    System.out.println("RUN: xy2links");
    Gbl.createConfig(args);

    System.out.println("  reading the network...");
    NetworkLayer network = null;
    NetworkLayerBuilder.setNetworkLayerType
        (NetworkLayerBuilder.NETWORK_DEFAULT);
    network = (NetworkLayer)World.getSingleton().createLayer
        (NetworkLayer.LAYER_TYPE,"false",null);
    NetworkParser network_parser = new NetworkParser(network);
    network_parser.parse();
    System.out.println("  done.");

    System.out.println("  setting up plans objects...");
    Plans plans = new Plans(Plans.USE_STREAMING);
    PlansWriter plansWriter = new PlansWriter(plans);
    plans.setPlansWriter(plansWriter);
    PlansReaderI plansReader = PlansReaderBuilder.getPlansReader(plans);
    System.out.println("  done.");

    System.out.println("  adding plans algorithm... ");
    plans.addAlgorithm(new XY2Links(network));
    System.out.println("  done.");

    System.out.println("  reading, processing, writing plans...");
    plansReader.read();
    plans.runAlgorithms();
    plansWriter.write();
    System.out.println("  done.");

    System.out.println("RUN: xy2links finished.");
    System.out.println();
}
```

```
////////////////////////////////////
// main
////////////////////////////////////

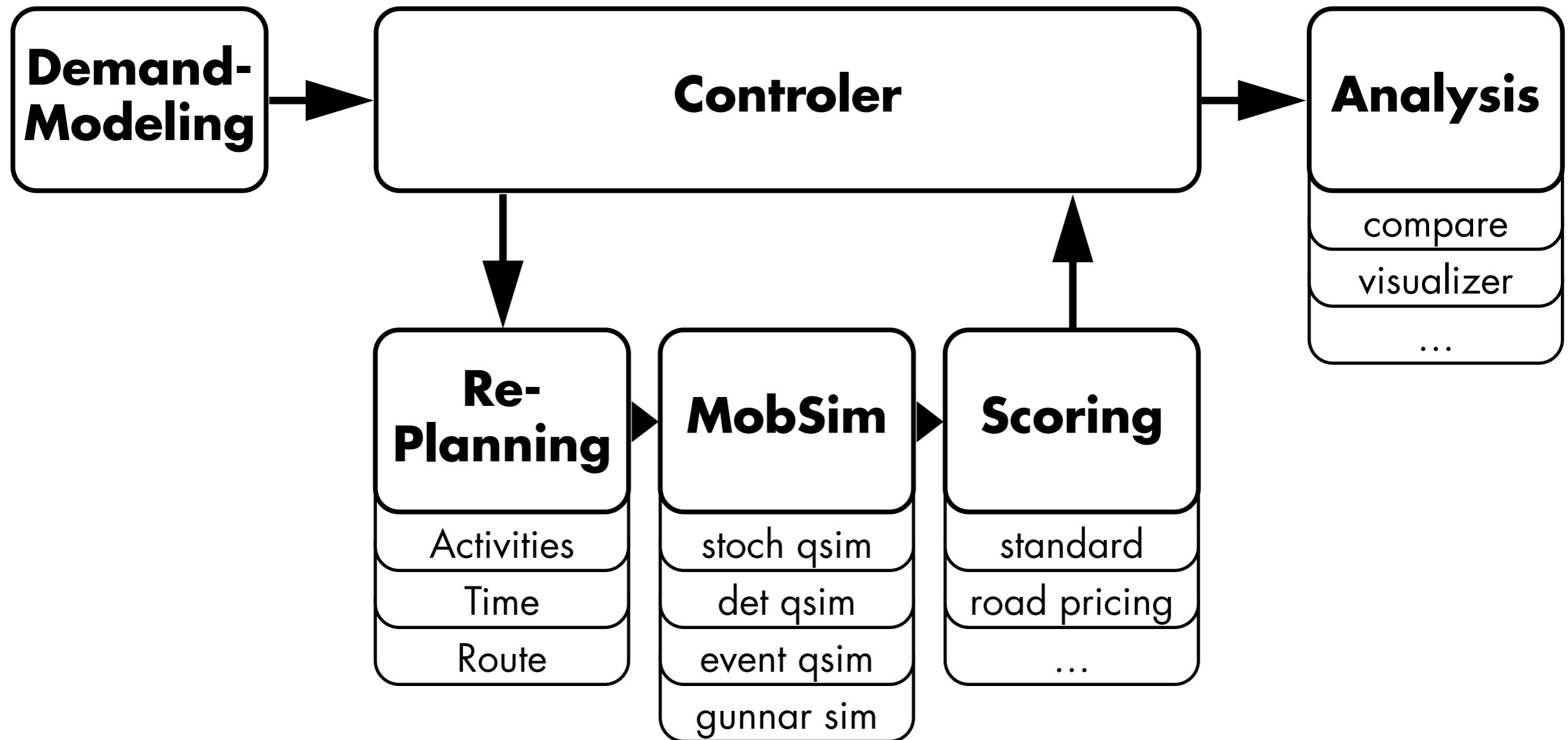
public static void main(String[] args) {

    /* ***  P L A N S  *** */
    // convertPlans(args);
    // filterSelectedPlans(args);
    // filterPlansInArea(args, 4580000, 5807000,
        4617000, 5835000); //berlin
    // removeLinkAndRoute(args);
    // filterCars(args);
    // filterWork(args);
    // filterWorkEdu(args);
    // xy2links(args);
    // calcRoute(args);
    // calcScore(args);

    /* ***  N E T W O R K S  *** */
    // convertNetwork(args);
    // renumberNetwork(args, false);
    // scaleCapacity(args, 22, 0.6, 1.2);
    // cleanNetwork(args);
    // calcRealSpeed(args);
    // calcNofLanes(args);

    /* ***  A N A L Y S I S  *** */
    // calcPlanStatistics(args);
    // analyzeLegDepTimes_plans(args, 300);
    // analyzeLegTimes_events(args, 300);
    // generateRealPlans(args);
    // planPlotActLocations(args);
    // writeVisumRouten_plans(args);
    // vehCountPerLinkViz(args);
    // calcODMatrices(args);
    // ... currently 45 routines available
}
```

Modules



Modules

Replanning modules

- Activities
- Time
- Route

Simulation modules

- different (mobility) simulations

Scoring modules

- different scoring algorithms

Analysis modules

- link travel time
- leg travel time
- agent travel time
- visualizer
- lot of other possibilities

Modules

Lots of useful modules, but...

- each module does something on its own
- modules often work as an algorithm over a collection
- modules do not (cannot?) offer their functionality to other modules
- code must be repeated in different locations
- can lead to inconsistencies, different implementations in different locations

Concrete example: planomat

- planomat has its own router
- planomat has its own scoring
- planomat has its own events parser, plans parser
- is planomat still usable if we introduce road pricing?

Example: Router

A router calculates the route between two locations in a network

- router should operate on legs, maybe acts

```

package org.matsim.demandmodeling.plans.algorithms;

public class PlansCalcRoute extends PersonAlgorithm {

    public PlansCalcRoute(NetworkLayer network, TravelCostI costCalculator,
        boolean calcSelectedOnly) {
        // ...
    }

    public void run(Person person) {
        int nofPlans = person.getPlans().size();
        for (int planId = 0; planId < nofPlans; planId++) {
            Plan plan = (Plan)person.getPlans().get(planId);
            if (plan.isSelected() || !calcSelectedOnly)
                handlePlan(plan);
        }
    }

    private void handlePlan(Plan plan) { ... }
}

```

operates on persons!

includes filter mechanisms!

Example: Router

```

package org.matsim.demandmodeling.plans.algorithms;

public class PlansCalcRoute extends PersonAlgorithm implements PlanAlgorithmI {

    public PlansCalcRoute(NetworkLayer network, TravelCostI costCalculator,
                           boolean calcSelectedOnly) {
        // ...
    }

    public void run(Person person) {
        int nofPlans = person.getPlans().size();
        for (int planId = 0; planId < nofPlans; planId++) {
            Plan plan = (Plan)person.getPlans().get(planId);
            if (plan.isSelected() || !calcSelectedOnly)
            handlePlan(plan);
        }
    }

    public void run (Plan plan) {
        handlePlan(plan);
    }

    private void handlePlan(Plan plan) { ... }
}

```

Example: Router

Improvements

- PlanAlgorithm1: other modules can call the router to just route one plan
 - should there be something like LegAlgorithm1, Router1?
 - where's the difference between a router-algorithm and a dijkstra-implementation?
- it still works as PersonAlgorithm to easily route complete population

But...

- we can no longer only route the selected plan of a person
 - general filters coming (based on work of Yu Chen):

```
plans.addAlgorithm(new SelectedPlanFilter(router));
plans.addAlgorithm(new WorstPlanFilter(router));
```
- functionality only available for integrated Java modules
 - planomat.exe still cannot use this router!
 - Could data/requests be exchanged between Java and C/C++ in real time without files? Stub-implementations as workshop project?

Questions Concerning Modularity

What modules can be identified in the process overview?

Which modules must be easy replaceable?

- replaceable in code when needed, or in config-file?
- identifier or classname in config-file?

Do modules need their own interfaces?

- currently, many of those modules are just PlansAlgorithms, PersonAlgorithms

How can modules offer their functionality in a general way to other modules?

How do other modules know about the existence / instances of other modules?

- world as registry?
- pass instances as parameters / hard-coded?

Future Topics

Questions Concerning the Toolbox

What can also be called a „module“?

- Junks of code offering some functionality that can be clearly separated from other code
- Global functionality / Singletons may also be modules

What and how many global objects do we need?

- currently existing global objects: Gbl, World, Config
- currently missing (imho): Log
- maybe missing in the future: Time (for simulations, replanning)

Where does parallelization take part?

- C-Version: parallelization began at Controller-level
- Java-Version: single modules can offer parallel versions
- Shared-Memory machines for parallelization (multi-threading)
 - most algorithms could be easily threaded, e.g. router

Logging

Currently available

- `Gbl.errorMsg()`, `Gbl.warnMsg()`, `Gbl.infoMsg()`, `Gbl.debugMsg()`
- `System.out.println()`

Problems with current situation

- automatic analysis not easy to do
- No default „layout“ or „format“ of log-message

Requirements for future logging-mechanism

- all log items should include a time stamp
- start, stop of modules, algorithms, iterations
- file-operations: open, close of files?
- simple analysis results, like average score at the end of an iteration
- more human-readable or machine-readable?
 - e.g. use `#` at beginning of line for human-readable statements

Logging

Log4J

- <http://logging.apache.org/log4j/docs/>
- clean API, not only in Java, but also C++, .Net, Perl, PHP, pSQL
- Apache License 2.0 — not compatible with GPL!

java.util.logging („JUL“)

- similar, but not as powerful as Log4J
- Java 1.4 functionality, no licensing problem
- can create XML-Logs

Do-It-Yourself Logging

- Log4J seems to be an overkill, may also hurt performance
- DIY: optimized for our needs
- extended version of `Gbl.errorMsg()`, `Gbl.warnMsg()` etc.
- convention on log-messages still needed, could be supported by own API

Replacing Modules

Hard-Coded

- `plans.addAlgorithm(new CalcShortestRoute(network, new TravelTimeCalculator(network)));`
- `plans.addAlgorithm(new CalcLongestRoute(network, new TravelTimeCalculator(network)));`

Identifier in config.xml

- `<param name="router" value="shortest" />`
- `String router = Config.getSingleton.getParam(„strategies“, „router“);`
`if (router.equals(„shortest“)) plans.addAlgorithm(new CalcShortestRoute(network));`
`else if (router.equals(„longest“)) plans.addAlgorithm(new CalcLongestRoute(network));`
`else throw new Exception(„unknown router-type!“);`

– only predefined functionality available

Classname in config.xml

- `<param name="router" value="org.matsim.router.CalcShortestRoute" />`
- `String classname = Config.getSingleton.getParam(„strategies“, „router“);`
`plans.addAlgorithm((PersonAlgorithm)InstanceCreator.newInstance(classname));`

+ can be extended by people who only use the jar-release

– classes have to be compiled manually, no automatic dependency