

Future directions

Kai Nagel
TU Berlin

Software engineering

This section contains some remarks about the software engineering *meta*-standard.

Software engineering

MATSim 2004

Collection of **C++ programs and diverse scripts** (approach inherited from TRANSIMS).

Quality control:

- DTDs for xml file formats.
- Some test cases.

But:

- Little test coverage
- Tests not automatic
- Vastly different programming standards throughout project.

Main challenge: Software engineering.

Fairly **monolithic Java pgm**; static access to many global variables; everybody who programmed w/ developer status.

⇐ assessment (at that time) that most *"external" users* (in particular transport engineers) *would not write code*.

Quality control:

- Automatic nightly tests
- Increasing test case coverage
- One programming standard throughout project

Main challenge: software engineering.

Increasingly **reduced (!)** core package

Access to core package preferably via clearly identified (and stable) interfaces

```
import org.matsim.api.*
(import org.matsim.core.api.*)
```

Additional functionality from **"modules"** with clearly identified maintainers.

External users do write code.

Increasing number of *"internal" users who are not "core" developers*. (*)

Main challenges now a mix between software engineering and conceptual.

→ Software increasingly needs to support conceptual work (e.g. (*)).

Sounds simple. But is not: Programming in "modules" is different from programming via the core. E.g:

```
person.getKnowledge()
```

vs.

```
getKnowledge( person );
```

Also, `person.<auto completion>` will no longer give you all the attributes of a person. :-)

Consider using MATSim with config files only.

Consider Java-coding initial demand only (see tutorial Gregor Lämmel; version with "basic" interfaces very soon)

[[eclipse]]

Consider Java-coding based on the "basic" interfaces only

Consider becoming a maintainer of a "module"

This section contains information about software design issues ...

... which are triggered by conceptual needs ...

... which, in my view, will "eventually" be implemented.

Conceptual → Software engineering

Events

→ essentially there

→ should be more standardized

At the heart of current MATSim: Any behavioral decision ultimately is only accepted if it increases the score. Challenges:

- **Consistency** between "modules" and "core".
- **Flexibility** so that alternative scoring approaches can be explored outside the core.

MATSim already has some auto properties:

- xy2links
- time allocation
- mode choice
- 2ndary act loca choice

Might want more.

- But which ones?

MATSim designed as large-scale planning tool (e.g. "all-of-Switzerland" = 8 million persons).

- Focus on **computational speed**.
- **Within-day replanning** a bit neglected.

Within-day replanning in principle easy to implement via "events".

Enough for first steps.

But: Computationally too slow when every traveller an event handler.

We need to find a **balance** between

... being **responsive** ...

and

... making/keeping MATSim **stable** ...

and

... **not overcommitting** ourselves.

Conceptual

Conceptual issues, ctd.

And of this ...

... we have seen some in this meeting

Conceptual issues

I think ...

... we now need to get away from a somewhat breathless "software implementation" mode ...

... and include a more **conceptually oriented mode**:

- policy-oriented real-world studies.
- work towards "understanding what the software does".
- Concentrating on what the software already *can* do rather than what it *could* do after (even) more implementation.

Conclusion

We have already gotten fairly far (I think).

We still have far to go (I think).

:~)

[[...!!]]

Thanks a lot for coming here.

Next MATSim User Meeting in Zürich.