Events, Event-Handlers

Events

Events are the only real output of MATSim.

Every other "output" is from an analysis based on events.

In order to analyze a simulation, understanding events is crucial.

MATSim provides code-infrastructure (Event-Handlers) to process events.

Events

Events consist of:

- time
- type
- additional key-value pairs

Events are typically created by the mobsim.

MATSim Extensions (e.g. Public Transport, Signals & Lanes, Car Sharing, ...) can create additional events.

In many cases, events appear in pairs.
(start/end activity, depart/arrive, enter/leave link, ...)

Type of Events

Person Events

	Event Type	Class Name
Agent starts an activity	actstart	ActivityStartEvent
Agent ends an activity	actend	ActivityEndEvent
Agent starts a trip	departure	PersonDepartureEvent
Agent ends a trip	arrival	PersonArrivalEvent
Agent enters a vehicle	PersonEntersVehicle	PersonEntersVehicleEvent
Agent leaves a vehicle	PersonLeavesVehicle	PersonLeavesVehicleEvent
Agent gets stuck in simulation	stuckAndAbort	PersonStuckEvent

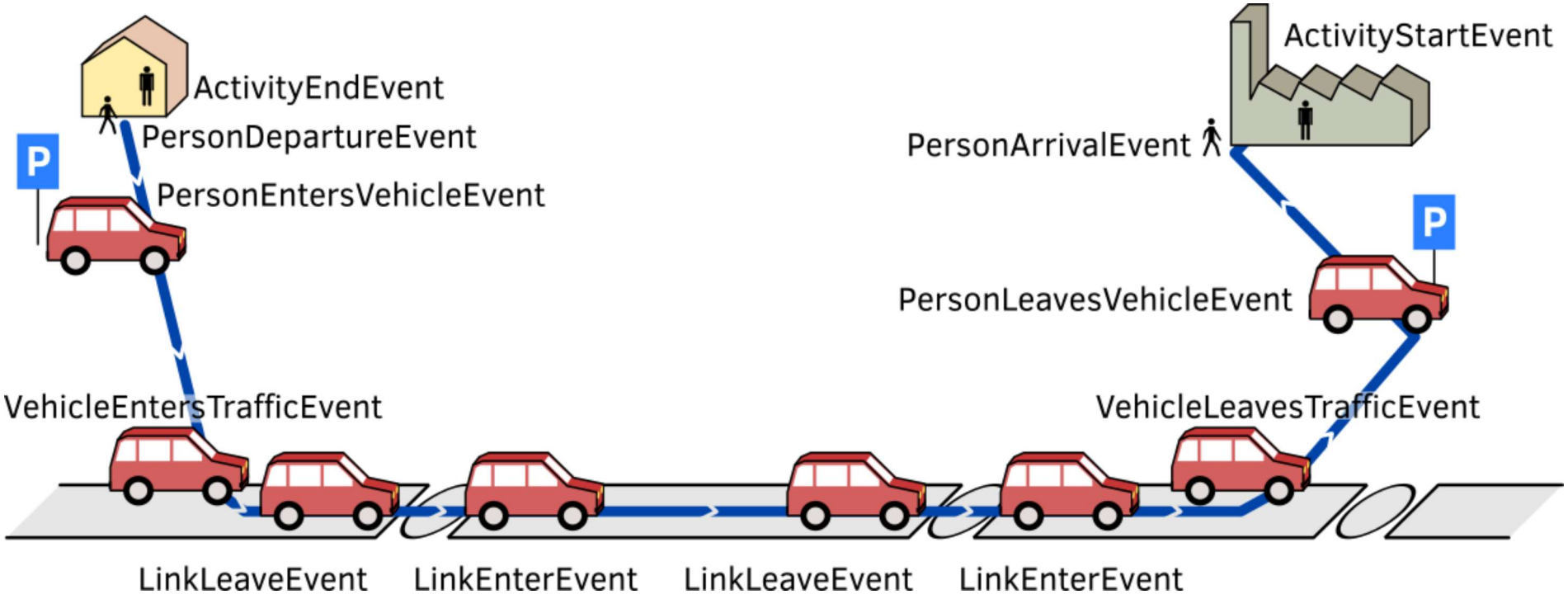
Type of Events

Vehicle Events

	Event Type	Class Name
Vehicle enters traffic (at departure)	vehicle enters traffic	VehicleEntersTrafficEvent
Vehicle leaves traffic (at arrival)	vehicle leaves traffic	VehicleLeavesTrafficEvent
Vehicle enters a link	entered link	LinkEnterEvent
Vehicle leaves a link	left link	LinkLeaveEvent
Vehicle gets stuck in traffic	vehicle aborts	VehicleAbortsEvent
PT vehicle arrives at a stop	VehicleArrivesAtFacility	VehicleArrivesAtFacilityEvent
PT vehicle departs at a stop	VehicleDepartsAtFacility	VehicleDepartsAtFacilityEvent

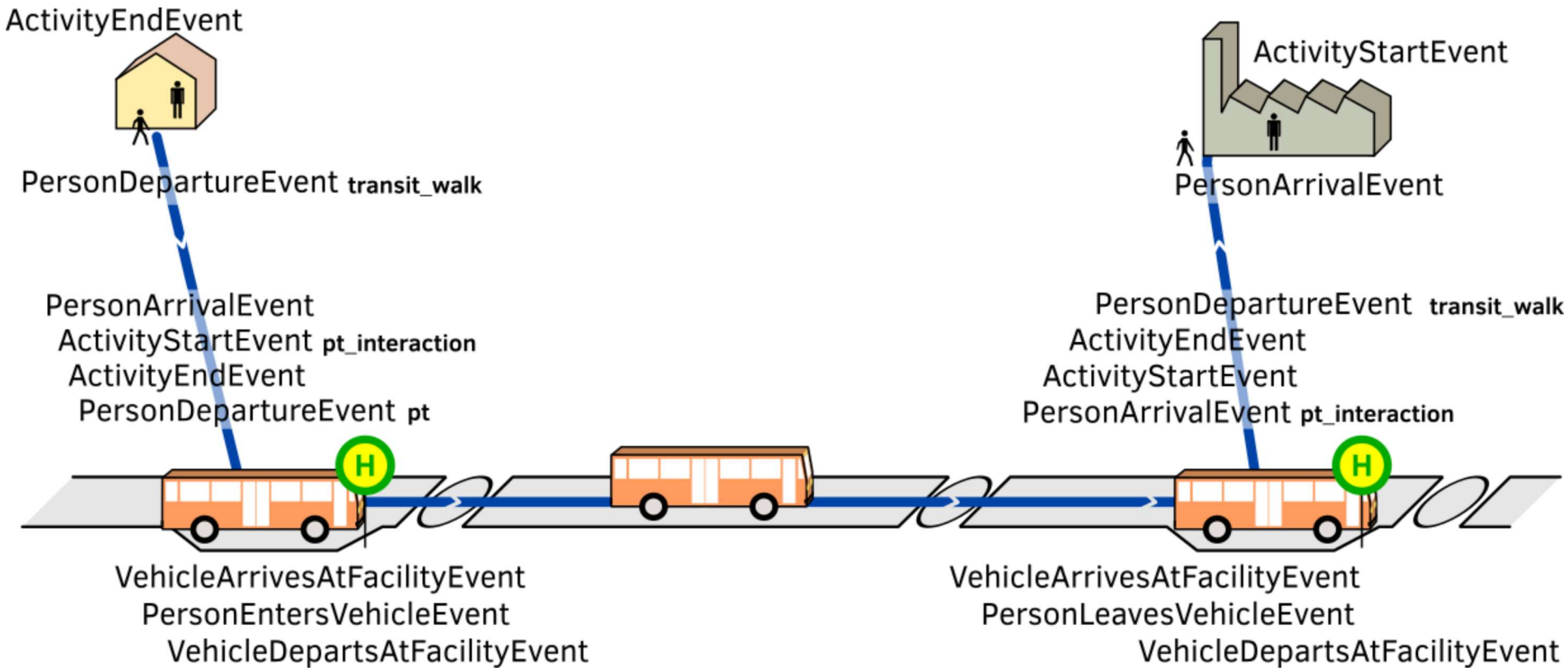
Typical Event Sequences

Car-Trip by an agent



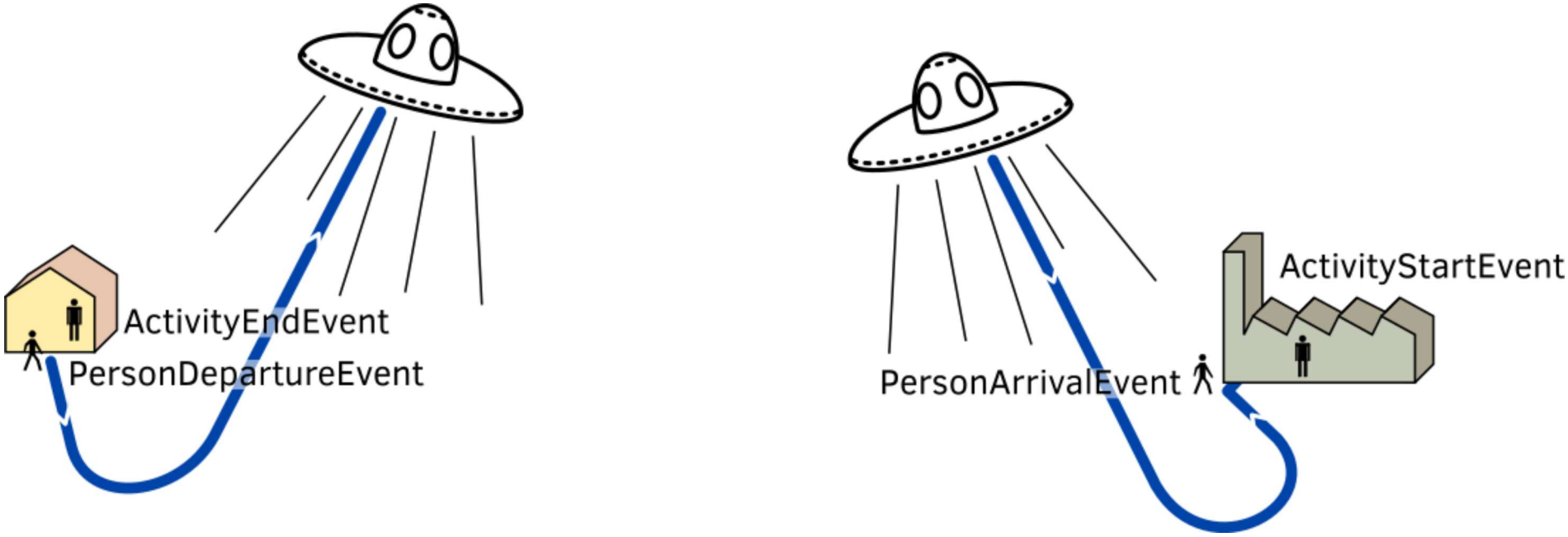
Typical Event Sequences

PT-Trip by an agent



Typical Event Sequences

Agent teleports



From Events to Analysis

Events are the major data source for analyses.

Network, Facilities, Public Transport schedule might be used for lookups.

Events should be processed as a stream, not as database table.

Processed/aggregated events can be stored in a database.

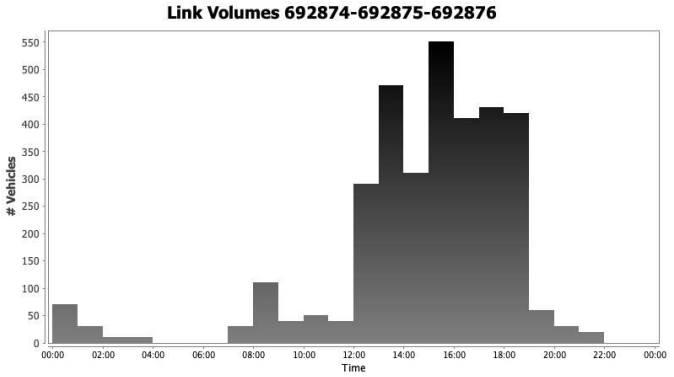
Example Analysis: Link Volumes

Static

Count all LinkLeave-Events, grouped by link-id

Dynamic

Count all LinkLeave-Events, grouped by link-id and hour



Example Analysis: Average durations

Average travel time (leg duration), by mode

- remember for each agent the departure time from a PersonDeparture-Event
- on each PersonArrival-Event, look up the departure time and calculate the duration
- add the duration to a sum for the specific mode, increase a counter for that mode
- after all events, calculate the average (sum / counter) for each mode

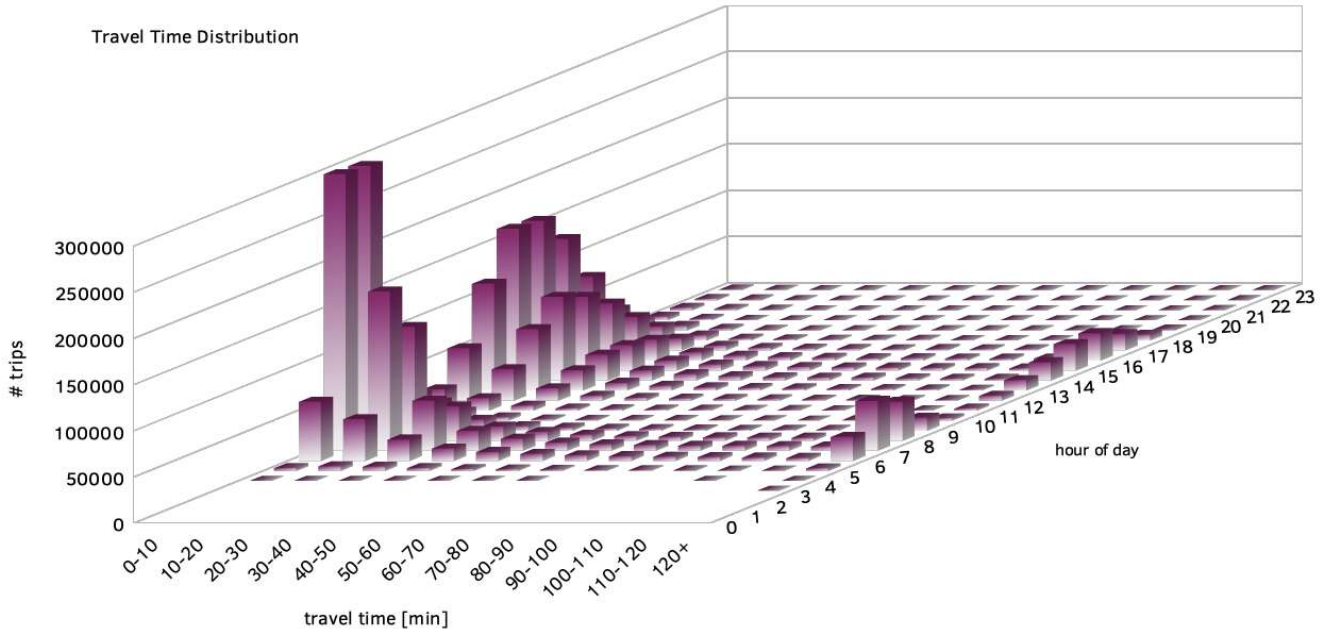
Average activity duration, by activity

- similar to avg. travel time above, but with ActivityStart- and ActivityEnd-Events
- Problem: the first activity only ends, the last one only starts
  - if they are from the same type, a wrap-around could be assumed and a duration calculated
  - otherwise assume midnight as start/end but that will create a bias
  - or just ignore these activities

Example Analysis: Travel Time Distribution

Travel time distribution, by hour of departure time

- similar to avg. travel time, but with more bins to sum the values up
  - e.g. a bin for departures between 07:00 and 08:00 and travel times between 10 and 20 minutes.



Example Analysis: Select Link Analysis

Often used by transport planners to see where travellers that travel on a specific link (the "selected link") come from and where they go to.

Example Analysis: Select Link Analysis

Idea

- Collect all vehicle trajectories that pass a specific link

Implementation

- Initialize for each link a separate counter
- For each starting vehicle, create an empty trajectory
- For each EnterLink-Event, add that link to the vehicle's trajectory
- For each arriving vehicle, take the trajectory and test, if it passed the link
  - if no: delete the trajectory
  - if yes: increase the counter for each link part of the trajectory

Alternative implementations are possible, some that require more than one pass over the events. E.g. first collect all vehicle ids passing the link, and then collect the trajectories of those vehicles only. (Time (Performance) vs. Memory-Consumption)

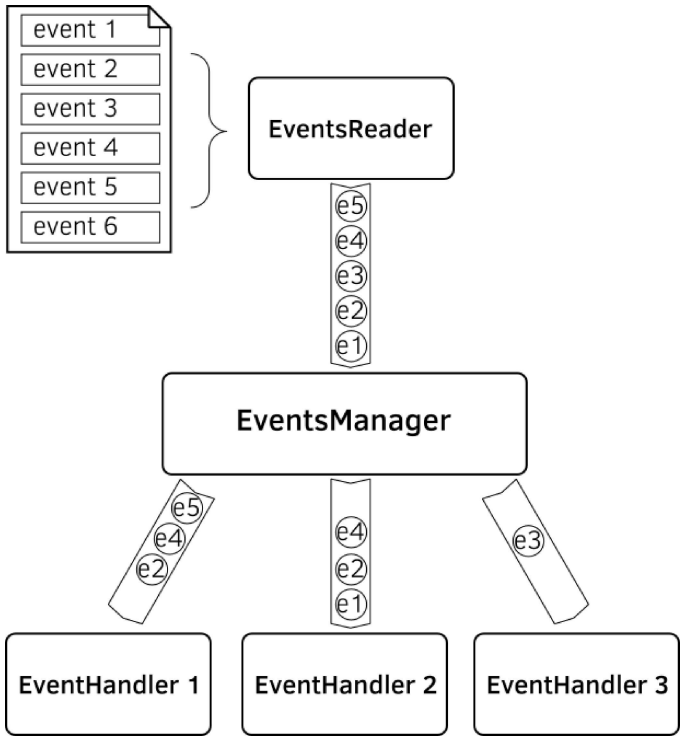
Processing an Events-Stream in MATSim

All events are sequentially processed by an EventsManager, which distributes the events to different EventHandlers depending on their type.

EventHandler
handles events of one (or more) types

EventsManager
keeps track of registered EventHandlers and distributes events accordingly

EventsReader
Reads all events from a file, and passes one after the other to the EventsManager.



EventHandlers

```
public class DepartureCounter implements PersonDepartureEventHandler {
  int count = 0;

  @Override
  public void handleEvent(PersonDepartureEvent event) {
    this.count++;
  }
}
```

```
public class WorkCounter implements ActivityStartEventHandler {
  int count = 0;

  @Override
  public void handleEvent(ActivityStartEvent event) {
    if ("work".equals(event.getActType())) {
      this.count++;
    }
  }
}
```

EventHandlers

```
public class AverageTravelTimeCalculator implements PersonDepartureEventHandler, PersonArrivalEventHandler {
  private final Map<Id<Person>, Double> departureTimes = new HashMap<>();
  private double travelTimeSum = 0.0;
  private int travelTimeCount = 0;

  @Override
  public void handleEvent(PersonDepartureEvent event) {
    this.departureTimes.put(event.getPersonId(), event.getTime());
  }

  @Override
  public void handleEvent(PersonArrivalEvent event) {
    double departureTime = this.departureTimes.get(event.getPersonId());
    double travelTime = event.getTime() - departureTime;

    this.travelTimeSum += travelTime;
    this.travelTimeCount++;
  }

  public double getAverageTravelTime() {
    return this.travelTimeSum / this.travelTimeCount;
  }
}
```

EventsManager: Post-Processing Events

```
EventsManager events = EventsUtils.createEventsManager();

MyEventHandler dataCollector = new MyEventHandler();

events.addHandler(dataCollector);

new MatsimEventsReader(events).readFile("/path/to/output_events.xml.gz");
```

Processing Events during the Simulation

It is possible to run EventHandlers during the simulation, instead of afterwards.

- Create statistics during simulation for calibration
- Collect traffic performance for within-day like functionality (e.g. car-sharing)

The same EventHandler instance will be reused for every iteration.
Add a `reset()` method to reset internal data structures between iterations:

```
public class DepartureCounter implements PersonDepartureEventHandler {
    int counter = 0;

  @Override
  public void reset(int iteration) {
      this.counter = 0;
  }

  @Override
  public void handleEvent(PersonDepartureEvent event) {
      this.counter++;
  }
}
```

Processing Events during the Simulation

Typically, EventHandlers only collect data.
This data needs further handling (e.g. writing values to output, write data to a file, …)

It is best to coordinate the EventHandler from a ControlerListener:

```
public class MyControlerListener implements StartupListener, IterationEndsListener {
  private final DepartureCounter depCounter = new DepartureCounter();

  @Override
  public void notifyStartup(StartupEvent event) {
    event.getServices().getEvents().addHandler(this.depCounter);
  }

  @Override
  public void notifyIterationEnds(IterationEndsEvent event) {
    log.info("# departures = " + this.depCounter.count);
  }
}
```

Processing Events during the Simulation

Enable the Controler Listener:

```
public static void main(String[] args) {
  String configFilename = args[0];
  Config config = ConfigUtils.loadConfig(configFilename);
  Scenario scenario = ScenarioUtils.loadScenario(config);
  Controler controler = new Controler(scenario);

  controler.addControlerListener(new MyControlerListener());

  controler.run();
}
```

Processing Events during the Simulation

Complex analyses might be run only every n-th iteration:

```
public static class MyControlerListener2 implements IterationStartsListener, IterationEndsListener {
  private final DepartureCounter depCounter = new DepartureCounter();

  @Override
  public void notifyIterationStarts(IterationStartsEvent event) {
    if (event.getIteration() % 10 == 0) {
      event.getServices().getEvents().addHandler(this.depCounter);
    }
  }

  @Override
  public void notifyIterationEnds(IterationEndsEvent event) {
    if (event.getIteration() % 10 == 0) {
      log.info("# departures = " + this.depCounter.count);
      event.getServices().getEvents().removeHandler(this.depCounter);
    }
  }

}
```

Useful Analysis: Trip Table

A trip table is an aggregation, listing each trip of an agent in a row.

It's a bit like a travel diary based on the simulation.

P_ID	T_ID	fromX	fromY	fromAct	depTime	toX	toY	toAct	arrTime	mode	...
123	1	425.2	972.5	home	08:14:48	1923.7	7329.7	work	08:37:13	car	...
123	2	1923.7	7329.7	work	17:23:29	744.1	824.6	shopping	17:59:03	car	...

Additional, useful columns:

- distance travelled
- coordinate of first/last pt stop (if it's a pt trip)
- access/egress walk distance (for pt trips)
- number of transfers (for pt trips)
- ...

Trip Table

Such a trip table would be a good candidate for a database.

A good starting point to build a trip table is in the analysis contrib: Events2TravelDiaries

Analyses

Try to implement an analysis on your own!

Ideas:

- link volumes
- intersection flow
- travel distances, total kilometers travelled
- trip table
- mode share (by trips or by distance)
- traffic jam detector
- ...

Model Calibration

Model Calibration

The goal of the calibration process is to have the model resemble reality as close as possible. (At least in those areas relevant for the model application.)

How could a model be "wrong"?

- unrealistic modal split
- too little/too much traffic on roads
- traffic peaks at the wrong time
- traffic jams at unexpected places
- too many/too few agents at certain places
- ...

Model Calibration

What can be calibrated in a model?

- Mode choice
- Location choice
- Route choice
- Departure time choice

Model Calibration

What can be calibrated in **MATSim**?

- **Mode choice**
- ~~Location choice~~
- **Route choice**
- **Departure time choice**

Location choice is typically part of the demand generation process.

Model Calibration

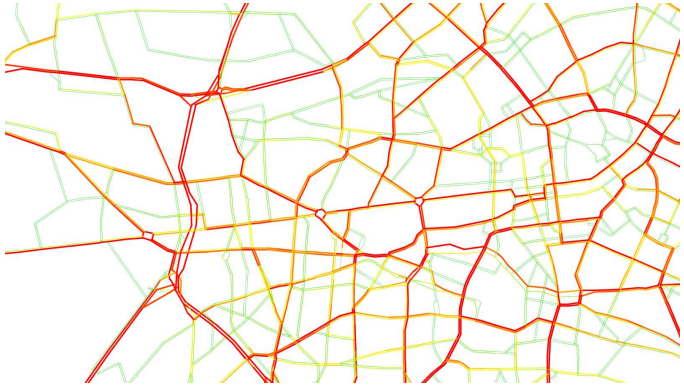
To calibrate, you need data to compare against.

Data availability is often dependent on the modelled region.

Calibration is (similar to demand modelling)
not standardized in MATSim.

One needs custom analyses to compare against
custom comparison data.

Calibration is basically done tweaking config parameters.



Model Calibration: Best Practices

Visual inspection

Look at the simulation in a visualizer.

- Do you observe traffic jams?
- Are they at the expected time and regions?

Modal Split

- Is the global modal split about what you expect?
- Has some mode a too high / too low share?

Make sure what you compare, e.g. legs vs. trips, urban areas vs rural areas vs global share.

Model Calibration: Best Practices

Car Counts

Compare link volumes to real world counting stations.

- Are morning and evening peaks at the right time?
- Are morning and evening peaks at the right amount?

MATSim provides functionality to compare against hourly car volumes.

PT Counts

Compare passenger-count at stations or along lines.

Model Calibration: Car Counts

Many road authorities use automated car counting stations with inductive loops embedded in roads.

Sometimes, this data is freely available. Often as hourly data for each day of a year.

We can compare such data with the traffic volumes simulated by MATSim.

Car Counts

MATSim provides a file format to specify the expected traffic volumes per hour on specific links:

```
<?xml version="1.0" encoding="UTF-8"?>
<counts xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://matsim.org/files/dtd/counts_v1.xsd"
        name="test" desc="test counting stations" year="2018">
   <count loc_id="100" cs_id="005">
      <volume h="1" val="300.0"></volume>
      <volume h="2" val="150.0"></volume>
      <volume h="3" val="20.0"></volume>
      <volume h="4" val="30.0"></volume>
      <volume h="5" val="136.0"></volume>
      <volume h="6" val="780.0"></volume>
      <volume h="7" val="1180.0"></volume>
      <volume h="8" val="1160.0"></volume>
      <volume h="9" val="920.0"></volume>
      <volume h="10" val="835.0"></volume>
      <volume h="11" val="744.0"></volume>
            ...
      <volume h="23" val="460.0"></volume>
      <volume h="24" val=390.0"></volume>
   </count>
</counts>
```

loc_id refers to the MATSim link in the network

cs_id (counting station id) can refer to some identifier of the original counting station

Car Counts

When aggregating real-world counts data to create a MATSim counts.xml:

- Only aggregate values from Monday to Friday.
  In rare cases even just Tuesday to Thursday.
- Exclude school holidays
- Exclude public holidays
- Exclude the lowest 5% and the top 5%, as they are likely outliers

Car Counts

counts.xml can either be used by MATSim or by Via.

MATSim:

```
<module name="counts" >
  <param name="inputCountsFile" value="/path/to/counts.xml" />
  <param name="outputformat" value="txt,kml" />

  <param name="countsScaleFactor" value="1.0" />

  <param name="writeCountsInterval" value="10" />
  <param name="averageCountsOverIterations" value="5" />
</module>

<module name="global">
  <param name="coordinateSystem" value="EPSG:25832" />
</module>
```

Via:

- add counts.xml
- create layer of type `Traffic Counts (Car)`

Car Counts: Common Problems

- Mapping of counting stations to links in the network is manual labour and error prone.
  Via can help (copy link id, show connections)
- Directions are switched (e.g. inbound/outbound directions).
  recognizable when looking at morning and evening peaks.
- We compare against a single value per link (per hour), although there might be a broad distribution as well in reality.

Model Calibration: Some recipes

Unexpected Traffic Jams

- Has the network the right capacities?
- Where do the agents come from, where do they go to?
    - Are too many going to a specific location?
    - Should they prefer another route?
    - Is the other route also congested?
    - Has the other route a coding-error in the network?
- Is the time distribution correct?

Select Link Analysis

Model Calibration: Some recipes

Wrong modal split

- Is mode choice enabled and correctly configured?
- Adapt the scoring parameters for this or the other modes.
- Has the network an unrealistic bottleneck?
- Is public transport unreliable?

Note that replanning randomly tells agents to try out alternative modes.
This can result in rarely used modes to be over-represented while innovation is active.
Compare against values from iterations with mode choice switched off.

```
<module name="strategy" >
  <param name="fractionOfIterationsToDisableInnovation" value="Infinity" />
</module>
```

Model Calibration: Some recipes

Wrong traffic peaks

- Are traffic peaks too early / too late?
- Are peak hours too low, and off-peak hours too high?

- Adapt opening times of activities to force agents to be earlier/later
- Adapt typical duration of activities.

Model Calibration: Some recipes

In general:

- Run multiple simulations in parallel with different parameters.
- Compare simulation outcomes.
- Deduct new, improved parameters
- Try to change as few parameters as possible at a time.

ETH Zurich is trying to automate and improve this process,
but it's currently research / work-in-progress.

Model Calibration: Cadyts

Developed by Gunnar Flötteröd: github.com/gunnarfloetteroed/java
MATSim integration: matsim/contribs/cadytsIntegration

Promises automatic calibration against counting stations.

Applies additional score terms on a per-plan base.

Results often in high correlation to counts.

Not applicable to case studies.

Opdyts might fix this by estimating/optimizing parameters.
Opdyts is still in development.