



# MATSim

mopsy-team



Krótkie przypomnienie...

...czyli co to jest ten

**MATSim**  
Multi-Agent Transport Simulation

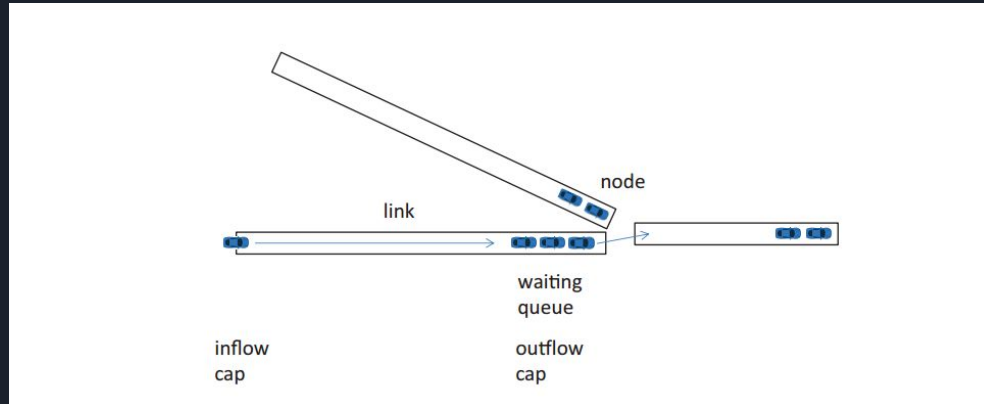


# MATSim - główne założenia

- mikroskopowy model ruchu i przeciążeń na drogach spowodowanych tym ruchem
- mikroskopowe modelowanie behawioralne - oparte na agentach i ich decyzjach
- sprawne, szybkie symulacje z dużą liczbą agentów
- zaawansowane, efektywne algorytmy wyboru optymalnych planów

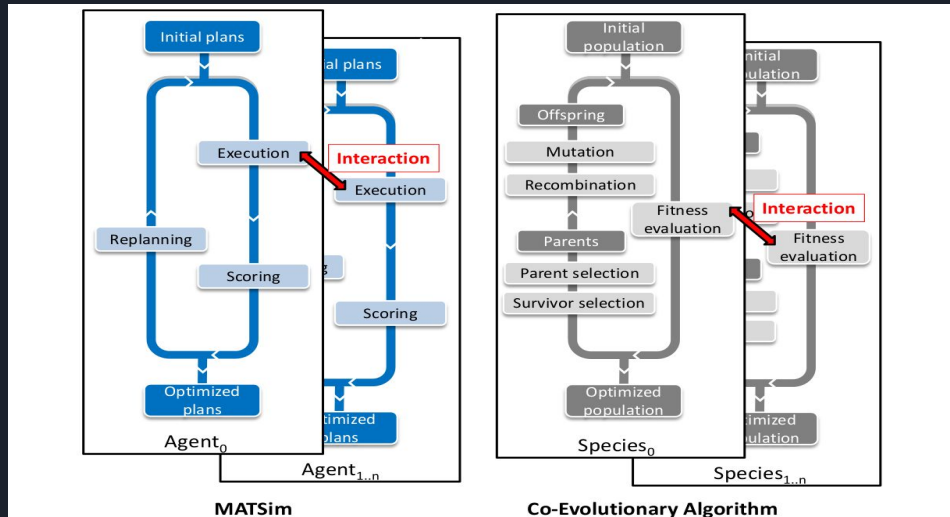
# Traffic flow w MATSimie

- symulacje oparte na kolejkach
- agent wjeżdżający na drogę jest dodawany na koniec kolejki
- czeka w niej, dopóki nie znajdzie się na jej początku i kolejny segment drogi nie pozwoli na wjazd
- przepustowość drogi i prędkość ruchu można określić w pliku konfiguracyjnym



# MATSim - algorytm koewolucyjny

- optymalizacja planów jest przeprowadzana indywidualnie przez każdego agenta
- inni agenci wpływają na jakość ich wykonania.
- w systemie ewolucyjnym dążylibyśmy do globalnego optimum





# mobsim - co to jest?

- mobility simulator
- moduł odpowiedzialny za przeprowadzanie symulacji
- MATSim posiada zaimplementowane dwa mobsimy - QSim i JDEQSim
- Popularny jest również DEQSim, napisany w C++



## JDEQSim

- implementacja DEQSim w Javie
- równoległa obsługa zdarzeń
- brak równoległej symulacji
- brak wspierania świateł drogowych
- brak możliwości uwzględnienia transportu publicznego
- przedziały czasu zależne od zdarzeń
- agenty są dotykane tylko gdy jest to wymagane (np. wjazd na skrzyżowanie ze światłami, staniecie w korku)



# QSim

- domyślny symulator używany w MATSimie
- pozwala na wielowątkowe przeprowadzanie symulacji
- oparty na przedziałach czasu
- w każdym przedziale czasu
- wiele pasów ruchu, światła



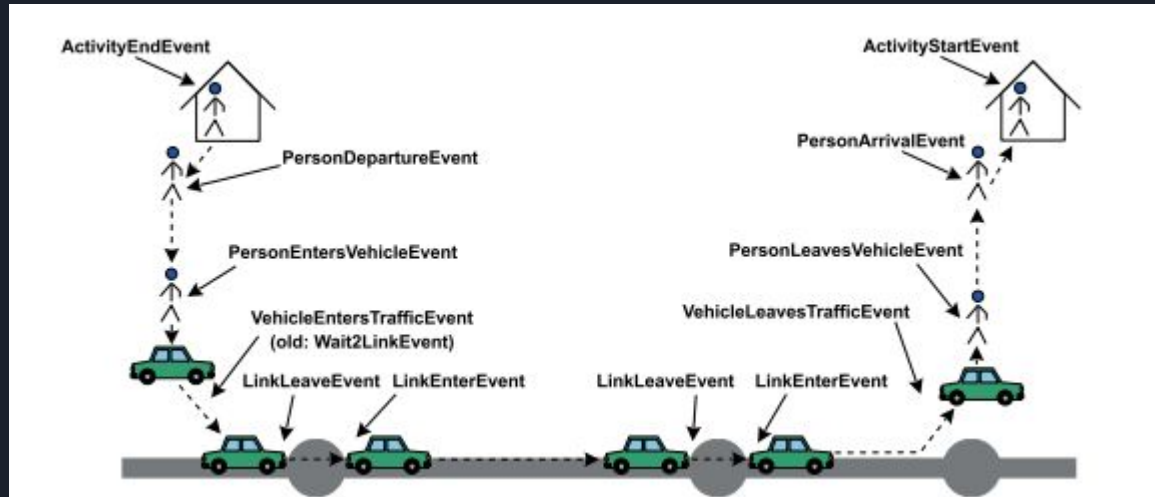


# QSim - rozróżnialność pojazdów

- każdy pojazd ma swoje ID
- kontrolowanie floty miejskiej na bieżąco
- ograniczenie - urealnienie połączeń pomiędzy transportem publicznym i prywatnym
- możliwość przemieszczania pojazdów przez różne osoby
- należy nadawać uprawnienia do korzystania z pojazdów
- system punktacji
- wprowadzanie poruszania się agentów pieszo

# MATSim - dane wyjściowe


- statystyki wyników dla poszczególnych agentów w kolejnych iteracjach
- statystyki odcinków drogowych
- dystanse
- zbiory zdarzeń
- czasy podróży





# Jakie funkcjonalności MATSima wydają się przydatne w kontekście MOPów?

- wyliczanie obciążenia na konkretnych odcinkach dróg
- znajdowanie zbyt długich czasów przejazdu danymi odcinkami
- możliwość porównania oceny sumarycznego zadowolenia podróżnych (np. przed i po dodaniu danej drogi przez inżyniera GDDKiA)
- (być może) efektywne projektowanie wjazdów, zjazdów z parkingów, infrastruktury parkingu



# Jakie rozszerzenia były stosowane do MATSima? (są w użyciu lub papiery na ten temat)

- drogi płatne - można uwzględnić przy ocenie danego planu
- taxi, komunikacja miejska
- auta elektryczne, stacje ładowania
- "Car-sharing"
- badanie emisji spalin
- plany ewakuacji



# MATSim - konfiguracja

3 główne pliki konfiguracyjne:

- metadane o symulacji
- informacje o populacji
- informacje o sieci drogowej



# MATSim - metadane o symulacji

- w postaci pliku xml
- podstawowe informacje o tym, jak ma zachowywać się symulacja
- parametry w postaci nazwa - wartość
- parametry układane są w logiczne grupy, np. te dotyczące kontrolera (np. liczba iteracji), dotyczące mobsimu (np. liczba wątków), itp.
- domyślny plik konfiguracyjny:

```
java -cp matsim.jar org.matsim.run.CreateFullConfig fullConfig.xml
```



# MATSim - konfiguracja kontrolera

Kontroler jest niezbędnym modułem, by uruchomić MATSim. Jego parametry umieszcza się w pliku konfiguracyjnym:

```
<module name="controler" >  
  ...  
</module>
```

Ważne parametry:

- liczba iteracji
- wybrany mobsim
- ścieżka folderu do zapisu danych symulacji
- algorytm wyboru ścieżki (Dijkstra, FastDijkstra, A\*, FastA\*)



# MATSim - współbieżność

Współbieżność w MATSimie jest realizowana na kilku poziomach:

- globalna liczba wątków:

```
<module name="global" >  
  <param name="numberOfThreads" value="2" />  
  ...  
</module>
```

Wątki są używane w kilku miejscach - przede wszystkim do obliczania wyników i do ustalania strategii w poszczególnych iteracjach.





# MATSim - współbieżność

- współbieżna obsługa zdarzeń:

```
<module name="global" >
  <param name="numberOfThreads" value="2" />
  ...
</module>
```

W MATSimie można obsługiwać poszczególne eventy produkowane przez mobsim. Oprócz liczby wątków, można również określić, czy chcemy je synchronizować i wprowadzić dane pomocnicze.



# MATSim - współbieżność

- współbieżny QSim:

```
<module name="qsim" >  
  <param name="numberOfThreads" value="10" />  
  ...  
</module>
```

Wielowątkowość wykorzystywana bezpośrednio w symulacji.



# MATSim - konfiguracja współrzędnych

- współrzędne sferyczne niezalecane - skomplikowane obliczanie odległości
- warto korzystać z systemów opartych na kartezjańskim układzie współrzędnych, często definiowanych dla poszczególnych regionów lub krajów
- najłatwiejszym sposobem określenia współrzędnych jest użycie kodów [EPSG](#):

```
<module name="global">  
  <param name="coordinateSystem" value="EPSG:32608" />  
</module>
```



# MATSim- konfiguracja współrzędnych

- Możliwe jest określenie innych współrzędnych w pliku konfiguracyjnym sieci drogowej - w takim przypadku współrzędne są najpierw konwertowane na te używane w symulacji:

```
<module name="network">  
  ...  
  <param name="inputCRS" value="EPSG:12345" />  
</module>
```



# MATSim - konfiguracja QSim

```
<module name="qsim" >  
  ...  
</module>
```

## Najważniejsze parametry QSima:

- numberOfThreads - liczba wątków
- flowCapacityFactor i storageCapacityFactor - parametry skalujące populację i pojemność poszczególnych odcinków dróg
- trafficDynamics - parametr określający zachowanie pojazdów podczas natężenia ruchu
- stuckTime - parametr określający po ilu sekundach bez ruchu pojazd jest usuwany lub (w zależności od konfiguracji) przenoszony - pozwala uniknąć deadlocków



# MATSim - konfiguracja QSim

QSim pozwala również zdefiniować różne typy pojazdów używanych w symulacji:

```
<module name="qsim">  
  <param name="mainMode" value="car,truck,bicycle" />  
  ...  
</module>
```

QSim można skonfigurować tak, by poszczególne pojazdy różniły się zachowaniem i produkowanymi eventami. Jeżeli w planie wystąpi inny tryb niż wymienione w pliku konfiguracyjnym, domyślnym zachowaniem jest teleportacja.

# MATSim - dane o sieci drogowej

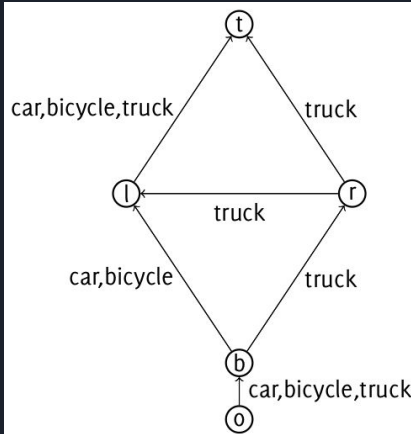
- określa infrastrukturę, po której poruszają się pojazdy
- składa się z wierzchołków i krawędzi

```
<network name="example network">
  <nodes>
    <node id="1" x="0.0" y="0.0"/>
    <node id="2" x="1000.0" y="0.0"/>
    <node id="3" x="1000.0" y="1000.0"/>
  </nodes>
  <links>
    <link id="1" from="1" to="2" length="3000.00" capacity="3600"
      freespeed="27.78" permlanes="2" modes="car" />
    <link id="2" from="2" to="3" length="4000.00" capacity="1800"
      freespeed="27.78" permlanes="1" modes="car" />
    <link id="3" from="3" to="2" length="4000.00" capacity="1800"
      freespeed="27.78" permlanes="1" modes="car" />
    <link id="4" from="3" to="1" length="6000.00" capacity="3600"
      freespeed="27.78" permlanes="2" modes="car" />
  </links>
</network>
```

# MATSim - dane o sieci drogowej

MATSim umożliwia również określenie trybów pojazdów, które mogą poruszać się poszczególnymi odcinkami dróg:

```
<module name="planscalcroute" >  
  <param name="networkModes" value="car, truck" />  
  ...  
</module>
```







# MATSim - dane o sieci drogowej

MATSim umożliwia również zmiany sieci drogowej w czasie. Określa się je w dodatkowym pliku konfiguracyjnym:

```
<param name="timeVariantNetwork" value="true" />
<param name="inputChangeEventsFile" value="path_to_change_events_file" />
```

Przykładowy element pliku networkChangeEvents.xml:

```
<networkChangeEvent startTime="03:06:00">
  <link refId="12487"/>
  <link refId="12489"/>
  <link refId="12491"/>
  <freespeed type="absolute" value="0.0"/>
</networkChangeEvent>
```



# MATSim - dane o populacji

- Każdy agent oddzielnie
- Dla każdego agenta określamy jego możliwe dzienne plany

```
<population>
  <person id="1">
    <plan selected="yes" score="93.2987721">
      <act type="home" link="1" end_time="07:16:23" />
      <leg mode="car">
        <route type="links">1 2 3</route>
      </leg>
      <act type="work" link="3" end_time="17:38:34" />
      <leg mode="car">
        <route type="links">3 1</route>
      </leg>
      <act type="home" link="1" />
    </plan>
  </person>
  <person id="2">
    <plan selected="yes" score="144.39002">
      ...
    </plan>
  </person>
</population>
```

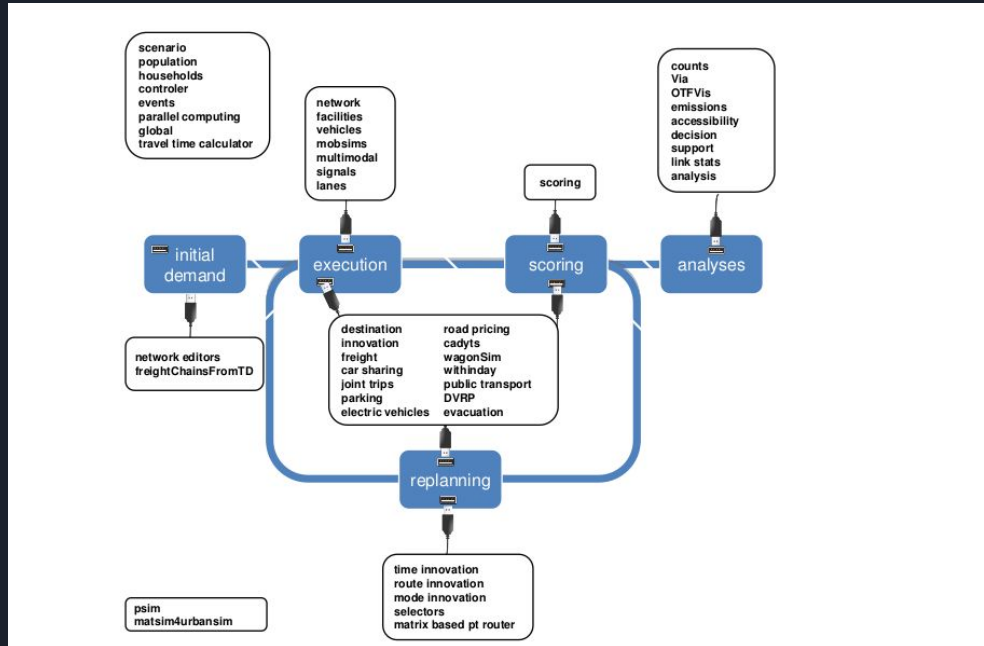


## MATSim - inne parametry

- teleportedModeParameters - obsługa teleportacji
- planCalcScore - parametry dotyczące obliczania wyników
- strategy - obsługa replanningu, strategię dotyczące wyboru, modyfikacji i usuwania planów
- counts - tworzenie wykresów porównujących dane godzinowe
- facilities - opcjonalny element - konfiguracja różnych punktów powiązanych z drogami
- vehicles - dane o flocie pojazdów
- households - dane o domostwach

# MATSim - funkcjonalność

MATSim podzielony jest na wielopoziomowe moduły:





# MATSim - poziomy dostępu

Z modularną budową MATSima wiąże się możliwość jego obsługi na kilku poziomach integracji:

- używanie tylko jądra MATSima i zdefiniowanych w nim funkcjonalności
- używanie oficjalnych rozszerzeń (ze strony <http://matsim.org/extensions>)



# MATSim - poziomy dostępu

- “Skrypty w Javie” - w tym przypadku używa się MATSima jako biblioteki, a Javy (możliwe jest także wykorzystanie innych języków) używa się do konfiguracji i obsługi zdarzeń.

```
... main( ... ) {  
    // construct the config object:  
    Config config = ConfigUtils.xxx(...) ;  
    config.xxx().setYyy(...) ;  
    ...  
  
    // load and adapt the scenario object:  
    Scenario scenario = ScenarioUtils.loadScenario( config ) ;  
    scenario.getXxx().doYyy(...) ; // (*)  
    ...  
  
    // load and adapt the controller object:  
    Controller controller = new Controller( scenario ) ;  
    controller.doZzz(...) ; // (**)  
    ...  
  
    // run the iterations:  
    controller.run() ;  
}
```



# MATSim - poziomy dostępu

“Skrypty w Javie” mogą się okazać przydatne do:

- modyfikacji konfiguracji
- modyfikacji scenariusza
- modyfikacji kontrolera



# MATSim - poziomy dostępu

- Używanie i testowanie dodatkowych rozszerzeń - znajdujących się w sekcji contrib oficjalnego repozytorium, a także w innych miejscach
- pisanie własnych rozszerzeń





# MATSim - pisanie własnych rozszerzeń

Pierwszy pomysł - dziedziczenie po klasie Controller i nadpisanie metod odpowiedzialnych za wywołanie mobsimu, scoring lub replanning.

Niezalecane, jeśli chcemy integrować różne modyfikacje w jeden produkt - np. nie można łatwo zmodyfikować rozszerzeń:

PublicTransportController, EmissionsController,  
RoadPricingController i OTFVisController w narzędzie do wizualizacji emisji spalin przez busy na konkretnych typach dróg.



# MATSim - rozszerzanie Config i Scenario

MATSim operuje na wielu rodzajach obiektów, np. nodes, links, persons, vehicles, które mają różne atrybuty. Czasami chcielibyśmy jednak dodać nowe atrybuty obiektom - np. wiek dla klasy person. MATSim zapewnia kilka sposobów modyfikacji obiektów:

- niektóre klasy w MATSimie implementują interfejs Customizable:

```
<dataType>.getCustomAttributes.put("myAttribName", myAttribValue) ;
```

Informacje dodatkowe - niezapisywane w plikach konfiguracyjnych.



# MATSim - rozszerzanie Config i Scenario

- ObjectAttributes - dodatkowy kontener dopisujący dodatkowe atrybuty do obiektów, które posiadają id:

```
attrs.putAttribute( id, attrName, attrValue ) ;
```

- Attributable - interfejs wprowadzony, by wyeliminować braki Customizable (niezapisywanie do plików xml) i ObjectAttributes (dotyczy tylko obiektów posiadających id).

```
<dataType>.getAttributes().putAttribute( String attribute, Object value) ;  
Object value = <dataType>.getAttributes().getAttribute( String attribute ) ;
```



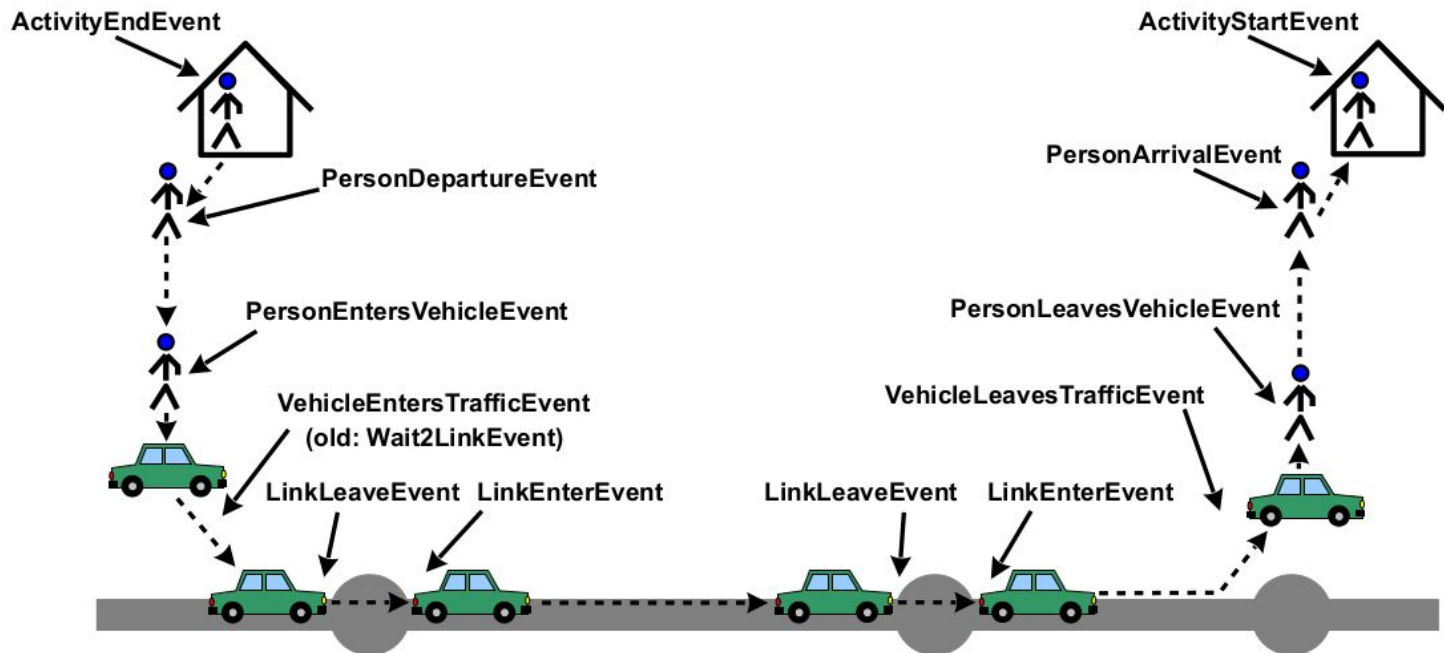
# MATSim - zdarzenia

Mobsim dokumentuje ruch pojazdów poprzez strumień zdarzeń. Zdarzenia to informacje o aktywności agentów w danym momencie. Przykład zdarzeń:

- agent zaczyna podróż
- agent wjeżdża /wyjeżdża z odcinka drogi
- agent przyjeżdża do celu

Każdy event zawiera znacznik czasu, typ i dodatkowe informacje z nim związane (t.j. id odcinka drogi, id pojazdu, itp.). W MATSimie przeprowadzano symulacje, które produkowały nawet kilka miliardów zdarzeń.

# MATSim - obsługa zdarzeń





# MATSim - obsługa zdarzeń

MATSim oferuje interfejs `EventHandler`, którego implementacja umożliwia własną obsługę danych produkowanych przez `mobsim`. Obiekty implementujące ten interfejs mogą być wykorzystywane przez całą symulację lub tak długo, jak trwa pojedyncza iteracja.

By produkować własne zdarzenia, można rozszerzyć klasę `Event`.



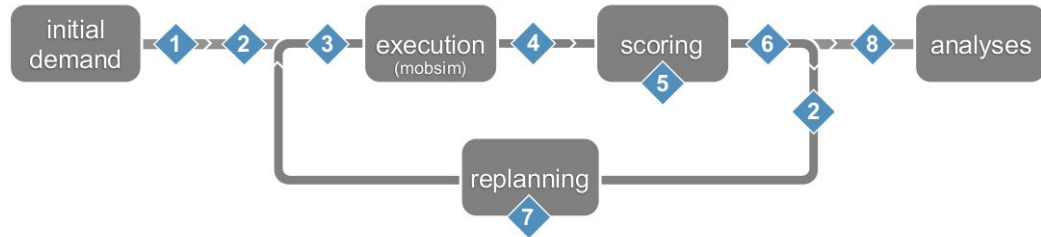
# MATSim - obsługa zdarzeń

Przykładowa obsługa zdarzenia może wyglądać następująco.  
Założmy, że chcemy analizować skręty w lewo:

- utworzenie klasy LeftTurnEvent
- utworzenie obiektu tej klasy za każdym razem, gdy samochód skręca w lewo
- może to być wykonane w klasie LinkEnterEventHandler - pojazd produkuje event przy każdym wyjeździe i wjeździe na odcinek drogi. Analiza dwóch kolejnych eventów tego typu i danych o sieci drogowej pozwala ustalić czy wykonany został skręt w lewo.

# MATSim - ControllerListeners

ControllerListeners to funkcje wywoływane w czasie obiegu MATSima. Mogą zostać wywołane w następujących momentach pętli:



## Controller Events:

- |                                 |                                |
|---------------------------------|--------------------------------|
| 1 Simulation Starts ("Startup") | 5 Scoring                      |
| 2 Iteration Starts              | 6 Iteration Ends               |
| 3 Before Mobsim                 | 7 Replanning                   |
| 4 After Mobsim                  | 8 Simulation Ends ("Shutdown") |





# MATSim - ControllerListeners

Przykładowy ControllerListener (wywołany zaraz po uruchomieniu) może wyglądać tak:

```
public class MyControllerListener implements StartupListener {  
    @Override  
    public void notifyStartup(StartupEvent event) {  
        ...  
    }  
}
```

Należy pamiętać, że porządek wywoływania Listenerów jest niezdefiniowany - nie należy dodawać dwóch Listenerów związanych z tym samym momentem pętli, w którym jeden korzysta z drugiego.



# MATSim - Controller Listeners

```
import org.matsim.core.controller.events.*;
import org.matsim.core.controller.listener.*;

public class MyIterationEndsListener implements IterationEndsListener {

    public void notifyIterationEnds(IterationEndsEvent event) {

        System.out.println("iteration " + event.getIteration()
            + " / " + event.getController().getLastIteration());

    }

}
```

---

```
public class MyController {

    public static void main(String[] args) {

        Controller c = new Controller(args);

        c.addControllerListener(new MyIterationEndsListener());

        c.run();

    }

}
```



# MATSim - Controller Listeners

```
public class MyEventsHandler implements ... {  
    ...  
    public void printStatistics() { ... }  
}
```

---

```
public class MyControllerListener  
    implements StartupListener, IterationEndsListener {  
    private MyEventsHandler eh = new MyEventsHandler();  
    public void notifyStartup(StartupEvent event) {  
        event.getController().getEvents().addHandler(this.eh);  
    }  
    public void notifyIterationEnds(IterationEndsEvent event) {  
        this.eh.printStatistics();  
    }  
}
```



# MATSim - mobsim listener

- Wywoływany w każdym kroku symulacji
- Może być użyty do wywołania Eventów, które nie są wywoływane przez inne Eventy
- Na przykład, w zależności od godziny możemy chcieć wywołać Eventy związane z pogodą
- Odpowiednie EventHandlery i MobsimListenery mogą działać działać równolegle - należy uważać z interakcją między nimi



# MATSim - TripRouter

- Obiekt posiadający funkcje generujące trasy pomiędzy lokacjami w zależności od osoby, trybu i godziny odjazdu
- Trip jest ciągiem elementów planu reprezentujących ruch
- Zwykle składa się z pojedynczego obiektu typu Leg lub kilku przedzielonych aktywnościami statycznymi (czyli bez poruszania się)
- RoutingModule jest domyślnym TripRouterem w MATSimie - klasę tę można rozszerzać lub modyfikować



# MATSim - alternatywny mobsim w Javie

- Oprócz dodawania MobsimListenerów, możliwe jest również całkowita podmiana mobsima
- Mobsim to klasa implementująca interfejs Runnable, która przyjmuje obiekt typu Scenario i produkuje strumień zdarzeń (obiektów Event)
- umożliwia to korzystanie z funkcji MATSima takich jak replanning i scoring, całkowicie zmieniając model ruchu



## MATSim - alternatywny mobsim w innych językach

MATSim posiada również pomocniczą klasę, która umożliwia wywołanie pliku wykonywalnego, który zapisuje strumień zdarzeń do pliku. Nie jest to jednak dobry sposób na przyspieszania obliczeń - potrzeba transferu danych między MATSimem a zewnętrznym mobsimem niweluje potencjalny zysk.



**KONIEC**