POLARIS – Motivations and Basics

Joshua Auld, Michael Hope

Hubert Ley, Vadim Sokolov, Bo Xu, Kuilin Zhang

Transportation Research and Analysis Computing Center

Argonne National Laboratory

June 5, 2013

Introduction

POLARIS

(Planning and Operations Language for Agent-based Regional Integrated Simulation)

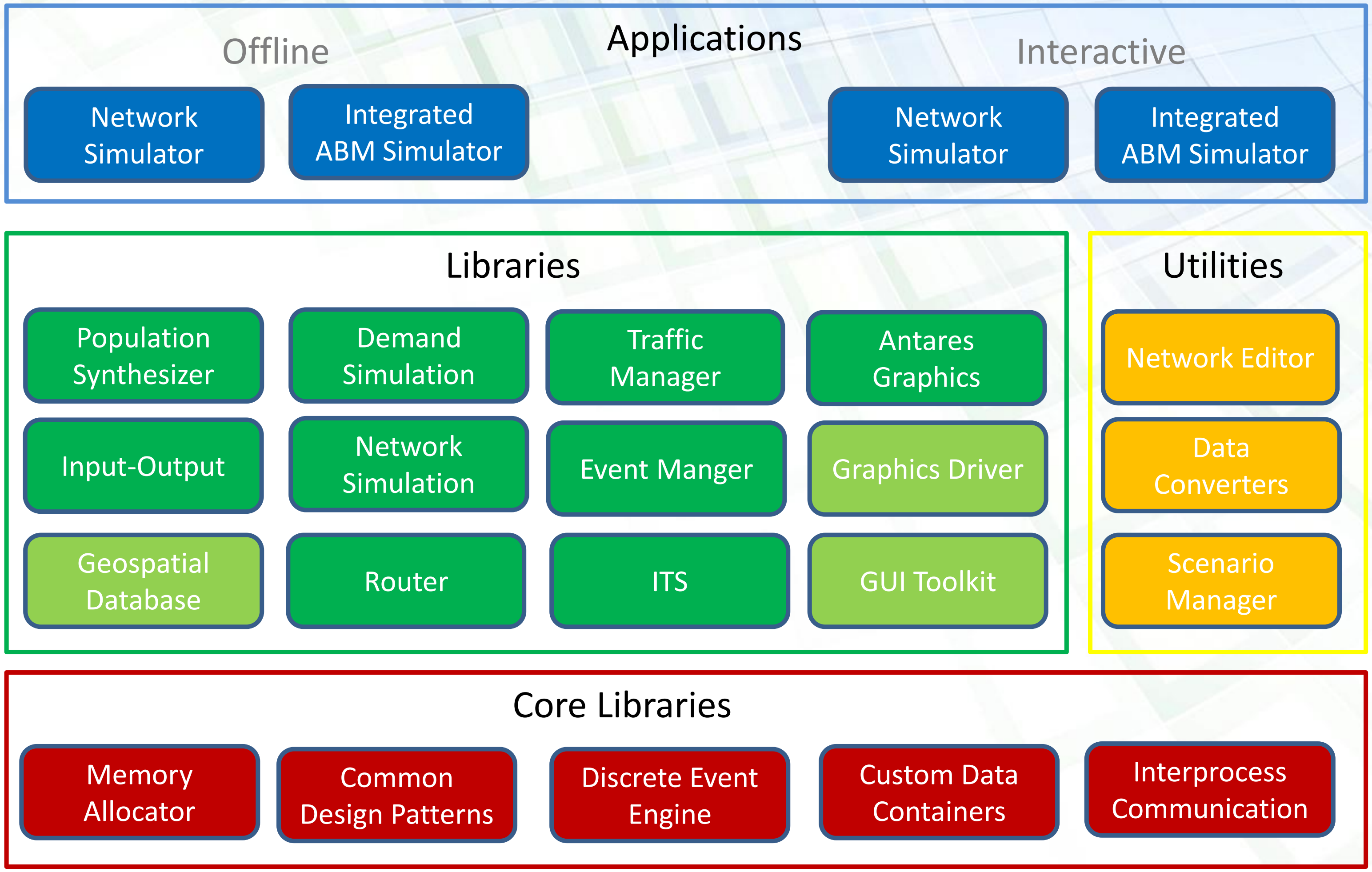
- Mandates from FHWA:
  1. Model Traffic Control Centers and other ITS Systems
  2. Enhance Interoperability among Existing Tools

- Core Goals and Philosophies of the POLARIS Effort:
  - Develop Transportation Modeling Standards and Protocols
  - Create an **Open Source** Model Development Environment
  - Seek Out Opinions from and Actively Listen to the Transportation Community
  - Connect Sub-Communities with a Common Modeling Framework
  - Offer Helpful Tools while Maintaining Flexibility and Modularity

What is POLARIS?

- General purpose core libraries which encourage flexible model development:
  - Modular, Extensible, Reusable
  - Agent Based
  - High Performance
- A repository of useful code and model fragments built using the core libraries
  - Common Transportation Objects
  - Open Source
  - Extended by Researchers
- Support utilities
- Fully developed applications:
  - Network Simulator
  - Integrated Activity Based Simulator

POLARIS Transportation Systems Modeling Suite



Agent Based Modeling

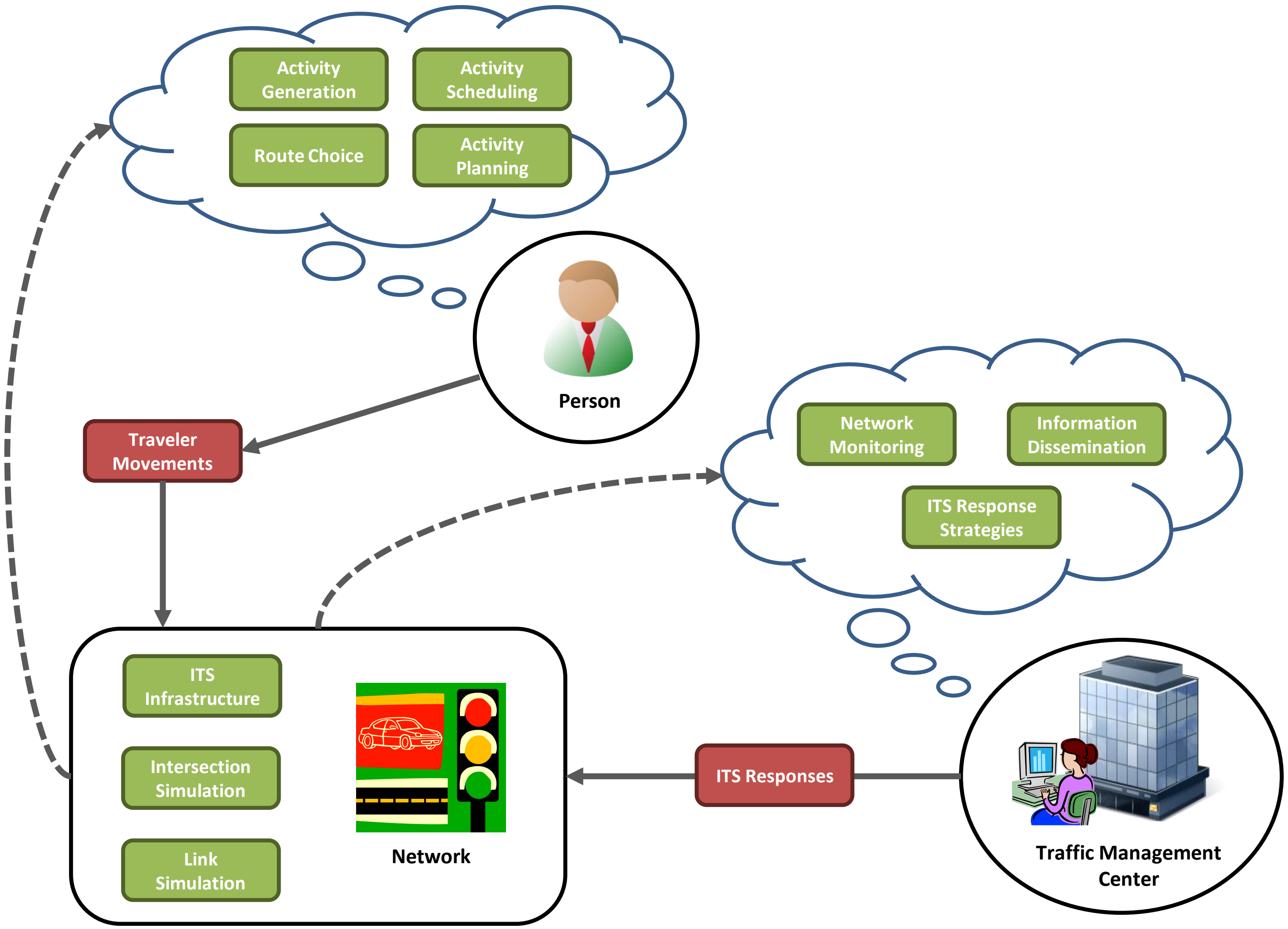
- Agent will encapsulate a set of behaviors that govern their interactions with other agents and with their environment

- Agent-based methodologies have proven to provide a structure that can be used to model a vast array of phenomena:
  - social processes
  - software systems
  - manufacturing systems
  - urban dynamics
  - economics

- In POLARIS, the agent-based paradigm is facilitated through a discrete event engine which puts developers in the "driver's seat" of each individual agent

- POLARIS core libraries and many extensions are not specific to transportation, they can be used for any model which can be simulated as an agent-based system

Agent Based Transportation Simulation System

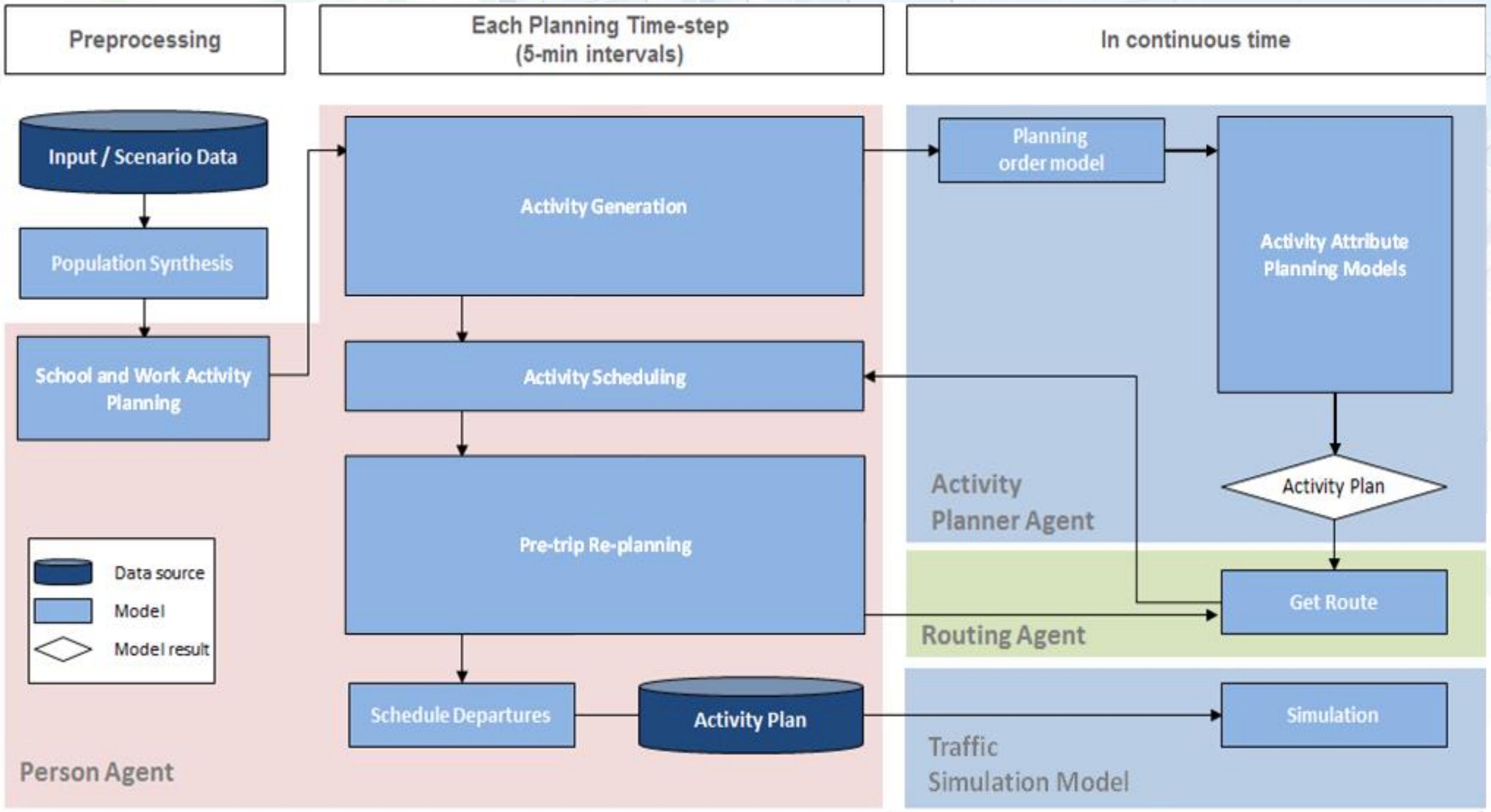


Activity-Based Travel Demand Model

Activity-Based Travel Demand Model

- Derived from ADAPTS activity-based model:
  - Simulation of how activities are planned and scheduled
  - Extends concept of “planning horizon” to activity attributes
  - Time-of-day, location, mode, party composition
- Fully agent-based travel demand model
  - All aspects of demand model are implemented as agent decisions
- Implemented in POLARIS language and adapted to discrete event engine
- **Core concept**:
  - Universal set of activity planning / scheduling processes represented by heuristics and/or models
  - Outcomes constrained by local context
  - Activity generation, planning, scheduling, etc. are events which are simulated

Activity-Based Travel Demand Model



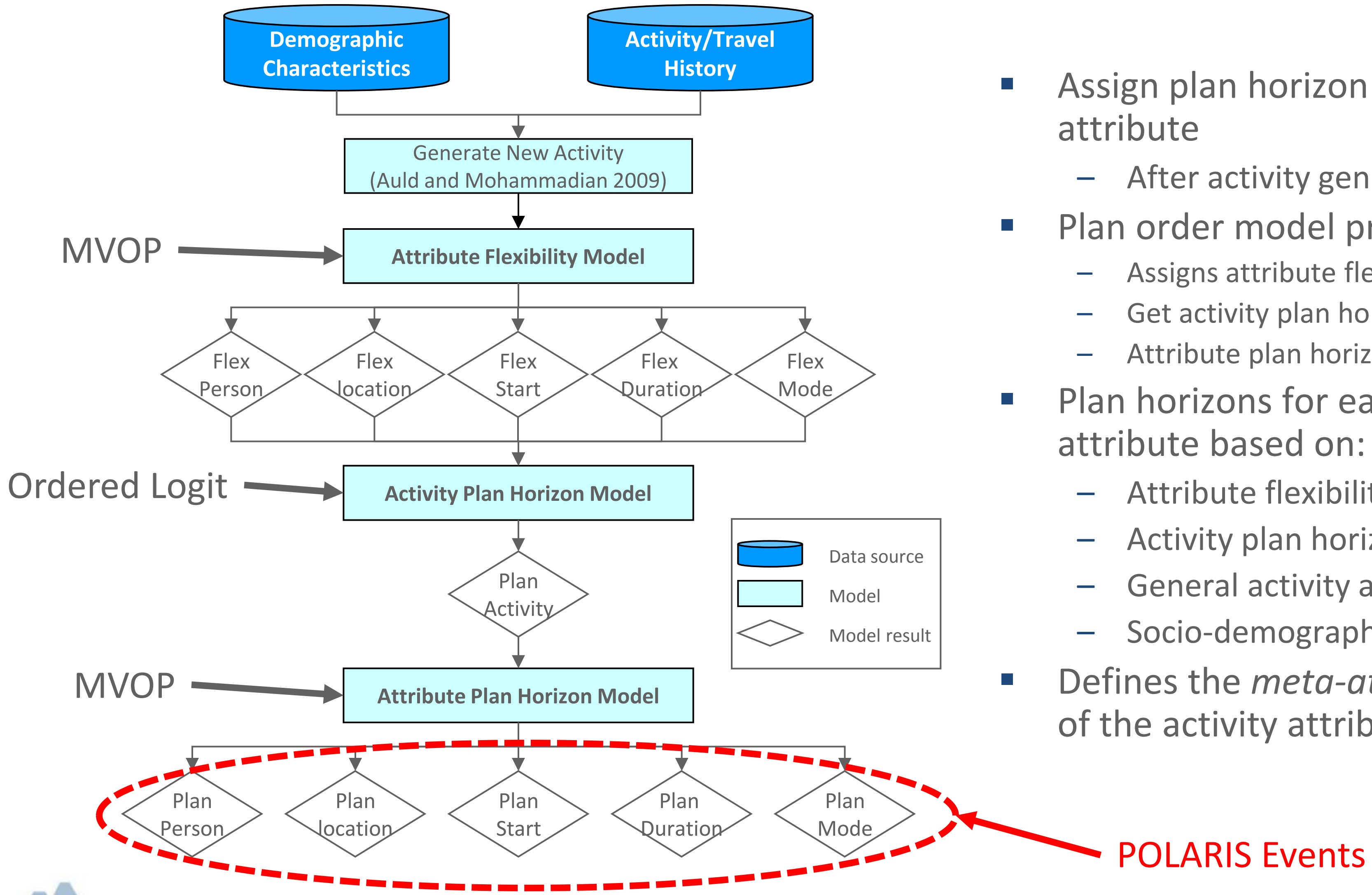
Activity Generation

- Activity generation with joint hazard-duration equations
  - Significant socio-economic variables
  - Impact of hazard rate from other activities
- Failure probability (generation) calculated each timestep
  - Based on time-since-last activity
  - Calculated using observed UTRACS and fit to CMAP survey through updating
  - $h_0 = \lambda_E \gamma_E (\lambda_E t)^{\gamma_E - 1} + \lambda_L \gamma_L (\lambda_L t)^{\gamma_L - 1}$: where $\gamma_L > 1$ and $\gamma_E < 1$
  - decreasing early failure (after trip chain), increasing late failure due to need growth over time

$$\begin{cases} h_i^{wev}(t_i, x_i, h_j^{nev}, h_k^{nev}, \ldots) = h_0^i \, e^{-(\beta_i x_i + \beta_{ji} h_j^{nev} + \beta_{ki} h_k^{nev} + \ldots)} \\ h_j^{wev}(t_j, x_j, h_i^{nev}, h_k^{nev}, \ldots) = h_0^k \, e^{-(\beta_k x_k + \beta_{ij} h_i^{nev} + \beta_{kj} h_k^{nev} + \ldots)} \\ h_k^{wev}(t_k, x_k, h_i^{nev}, h_j^{nev}, \ldots) = h_0^k \, e^{-(\beta_k x_k + \beta_{ik} h_i^{nev} + \beta_{jk} h_j^{nev} + \ldots)} \\ \quad \vdots \end{cases}$$

h^{wev} = hazard with exogenous hazard covariates
h^{nev} = without exogenous covariates

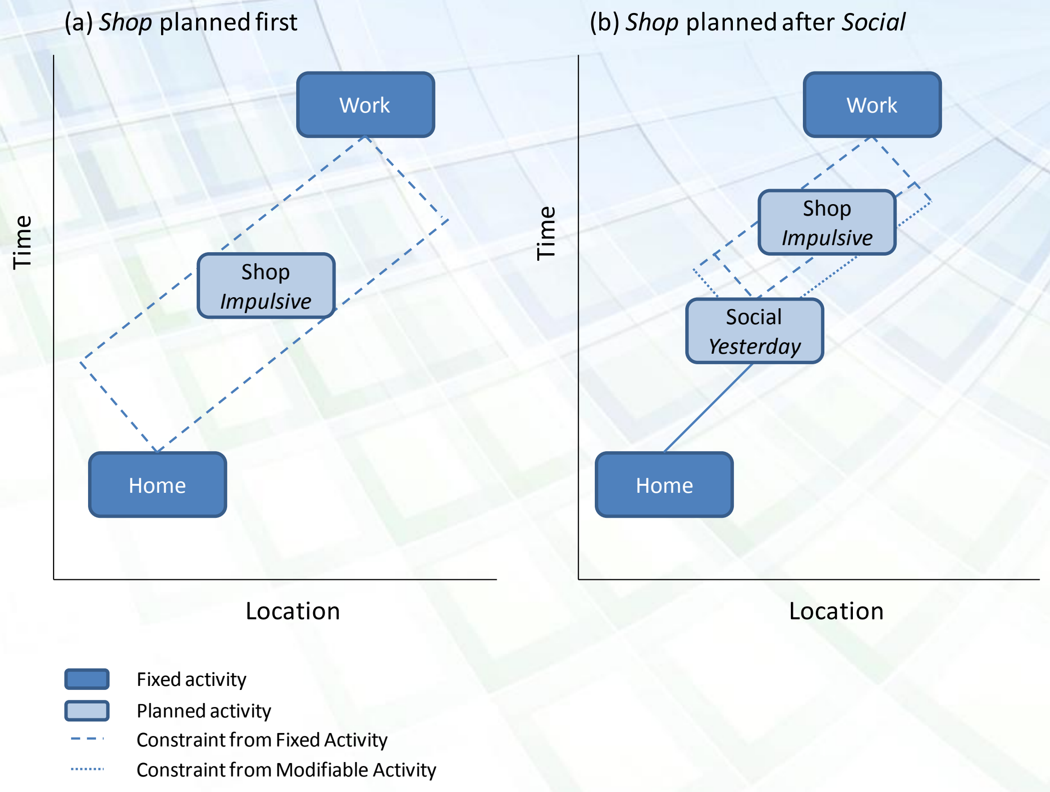
Planning Order Model



- Assign plan horizon to each attribute
  - After activity generated
- Plan order model process
  - Assigns attribute flexibility
  - Get activity plan horizon
  - Attribute plan horizons
- Plan horizons for each attribute based on:
  - Attribute flexibilities
  - Activity plan horizon
  - General activity attributes
  - Socio-demographics, etc.
- Defines the *meta-attributes* of the activity attributes

Planning Constrained Destination Choice

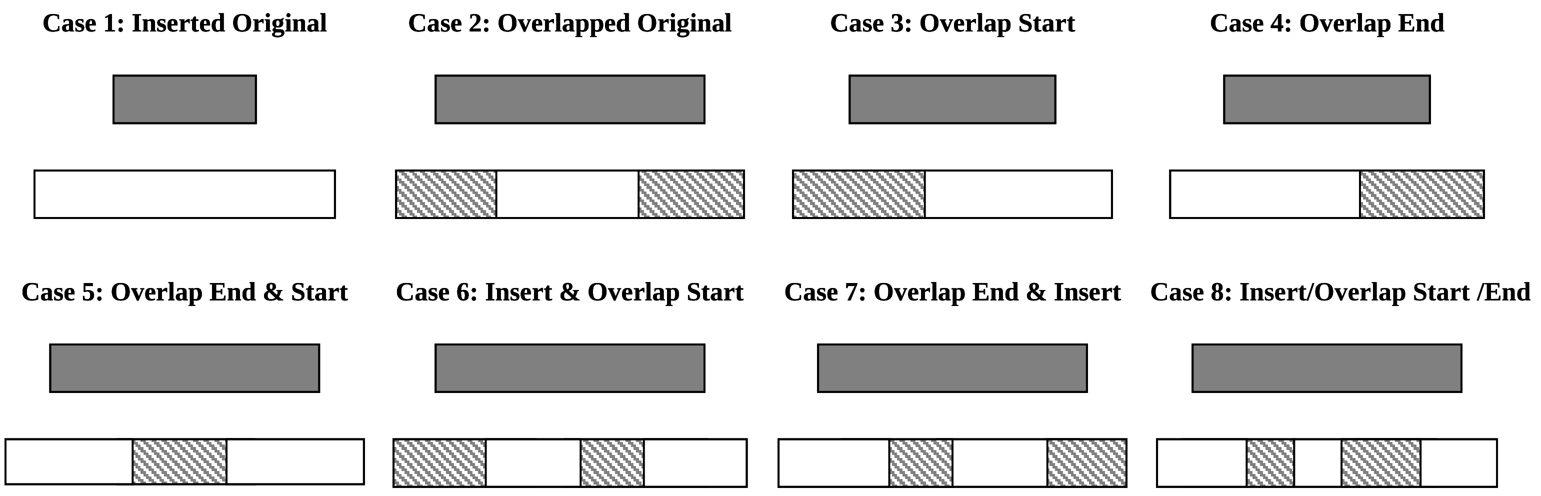
- Choose destination location from set of zones
  - Limit based on planned activities
  - Generates "Available Set" depending on Activity Plan Horizons
  - Requires plan horizon model to specify when activities planned
- Use *Stratified Importance Sampling* on "Available Set"



$$V_{in} = \overbrace{\beta_T T_{in}}^{\text{Impedance}} \overbrace{+ \beta_I ln(I_{in}) + \beta_R R_{in}}^{\text{Demographics}} + \overbrace{\gamma \ln\left(\sum_j^J \beta_j A_{ij} + \sum_k^K \beta_k E_{ik}\right)}^{\text{Attractiveness}} + \overbrace{\sum_k^K \theta_k C_k}^{\text{Competition}} + \overbrace{\ln\left(\frac{1}{p(i)}\right)}^{\text{Bias Correction}}$$

Activity Scheduling – Overall System

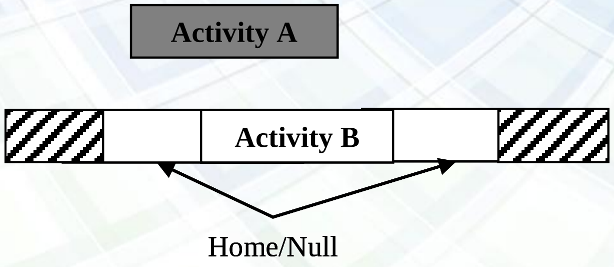
- Derived from TASHA (ILUTE) scheduling system
  - Activities added to schedule as simulation progresses
  - Overlaps happen due to planning process or schedule delay
  - Incorporates conflict resolution modeling
  - Resolution strategy determines which rules to follow
  - Resolution strategies and modification rules from observed empirical data
- When a new activity is added:
  1. Determines conflict type (shown below)
  2. Run conflict resolution model to determine resolution type
  3. Modify schedule to fit new activity based on resolution type
  4. Insert new activity or drop it (and restore original schedule)


Case 1: Inserted Original
Case 2: Overlapped Original
Case 3: Overlap Start
Case 4: Overlap End
Case 5: Overlap End & Start
Case 6: Insert & Overlap Start
Case 7: Overlap End & Insert
Case 8: Insert/Overlap Start /End

Activity Scheduling Rules

- In the scheduling rules, the situation would be handled as follows:
  1. If resolution type is ‘Delete Original’
     1. Remove Activity B from schedule, add Activity A
  2. If resolution type is ‘Modify Original’
     1. Move Activity B, align start of Activity B with end of Activity A
     2. Truncate Activity B
     3. Insertion is not feasible
  3. If resolution type is ‘Modify Conflicting’
     1. Move Activity A, align end of Activity A with start of Activity B
     2. Truncate Activity A
     3. Insertion is not feasible
  4. If resolution type is ‘Modify Both’
     1. Move Activity A, align end of Activity A with start of Activity B
     2. Move Activity B backward
     3. Truncate Activity A and Activity B proportional to durations;
     4. Insertion is not feasible.

Scheduling Example:

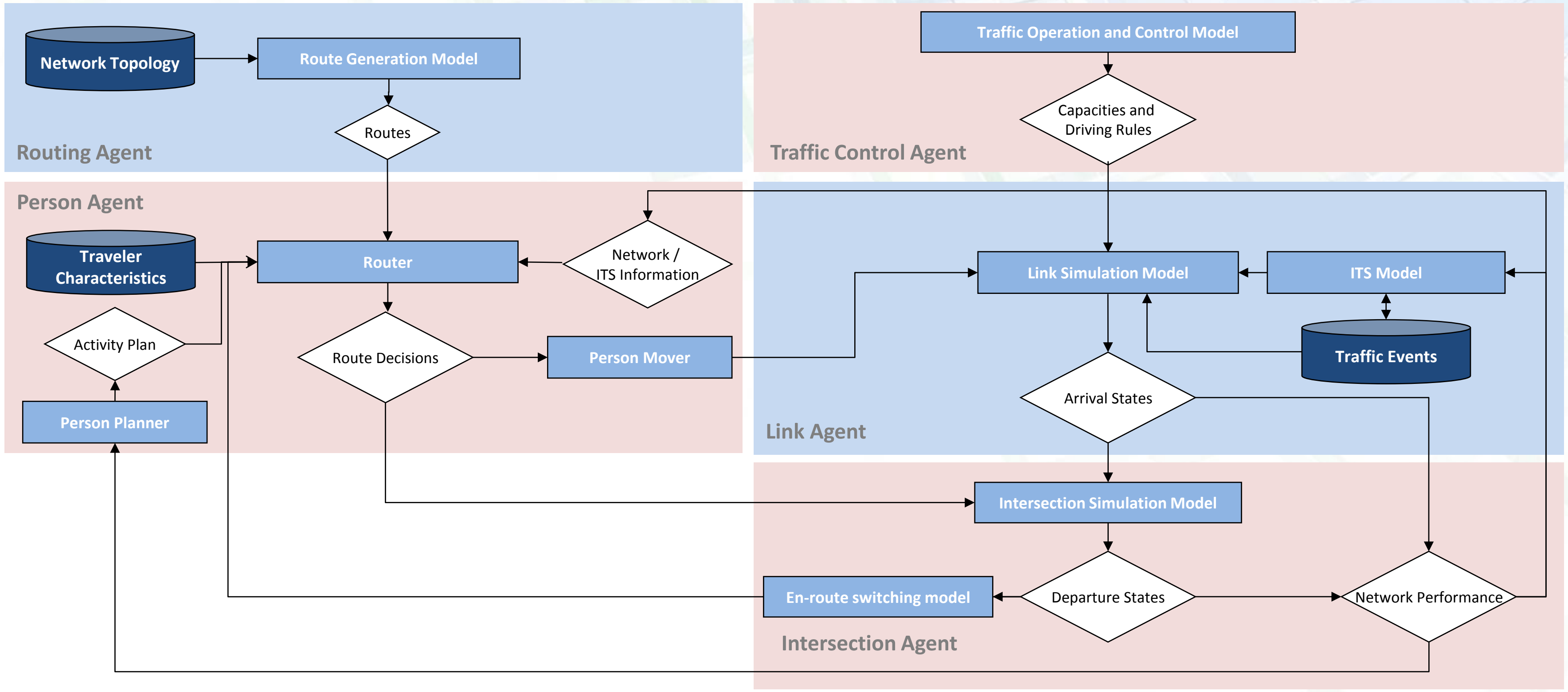


Simulation-Based Dynamic Traffic Assignment Model

Simulation-Based Dynamic Traffic Assignment Model

- Route Choice Model
- En-route Switching Model
- Traffic Control Model
- Mesoscopic Traffic Simulation Model
- Weather / Accident Reaction Model

Simulation-Based Dynamic Traffic Assignment Model



Route choice model

- **One-Shot Assignment** using prevailing travel information
- Averaged experienced travel times in last assignment interval (e.g. 5 minutes, user defined)
- Travel times are output from traffic simulation model
- Current implemented route choice model is **pre-trip route choice model**
  - **Pre-trip route choice model** is for pre-trip users who use the travel time information based on current traffic conditions to find a shortest path from his/her origin to destination. (e.g. using google map to compute shortest path considering traffic at that time)
  - **Shortest path algorithm**: link-based A-Star Algorithm that takes care of delay at turn movement

Traffic Simulation Model

- Newell's Simplified Kinematic Wave model
  - Using cumulative curves
  - Capturing queue formation, spillback, and dispersion
  - Capturing shock wave
  - Adhering to the fundamental diagrams
- Parameters
  - Simulation interval length (user defined, e.g. 6 seconds)
- Output
  - Network flow pattern
    - Cumulative vehicles at upstream and downstream of a link
    - Vehicle trajectory (enter time and exit time of each link)
  - Network performance
    - Time-dependent link travel time by turn movement (explicitly capture turn penalty)

Traffic Information Processing Model

- Prevailing travel time information
  - Average link travel times by turn movement using the experienced travel times in the previous assignment interval
    - E.g. for a 5 minutes assignment interval and 6 seconds simulation interval, there are 50 travel times in the previous assignment interval from the simulation model, hence, we average the traffic times using these 50 travel times for each link by turn movement

- Cumulative vehicles at upstream and downstream of each link
  - Can derive all other traffic variables such as inflow and outflow rates, density, queue length…
  - Can estimate any intermediate traffic condition between the upstream and downstream of the link using the **three-detector method** by Newell.

Intelligent Transportation System and Traffic Management Center Simulation

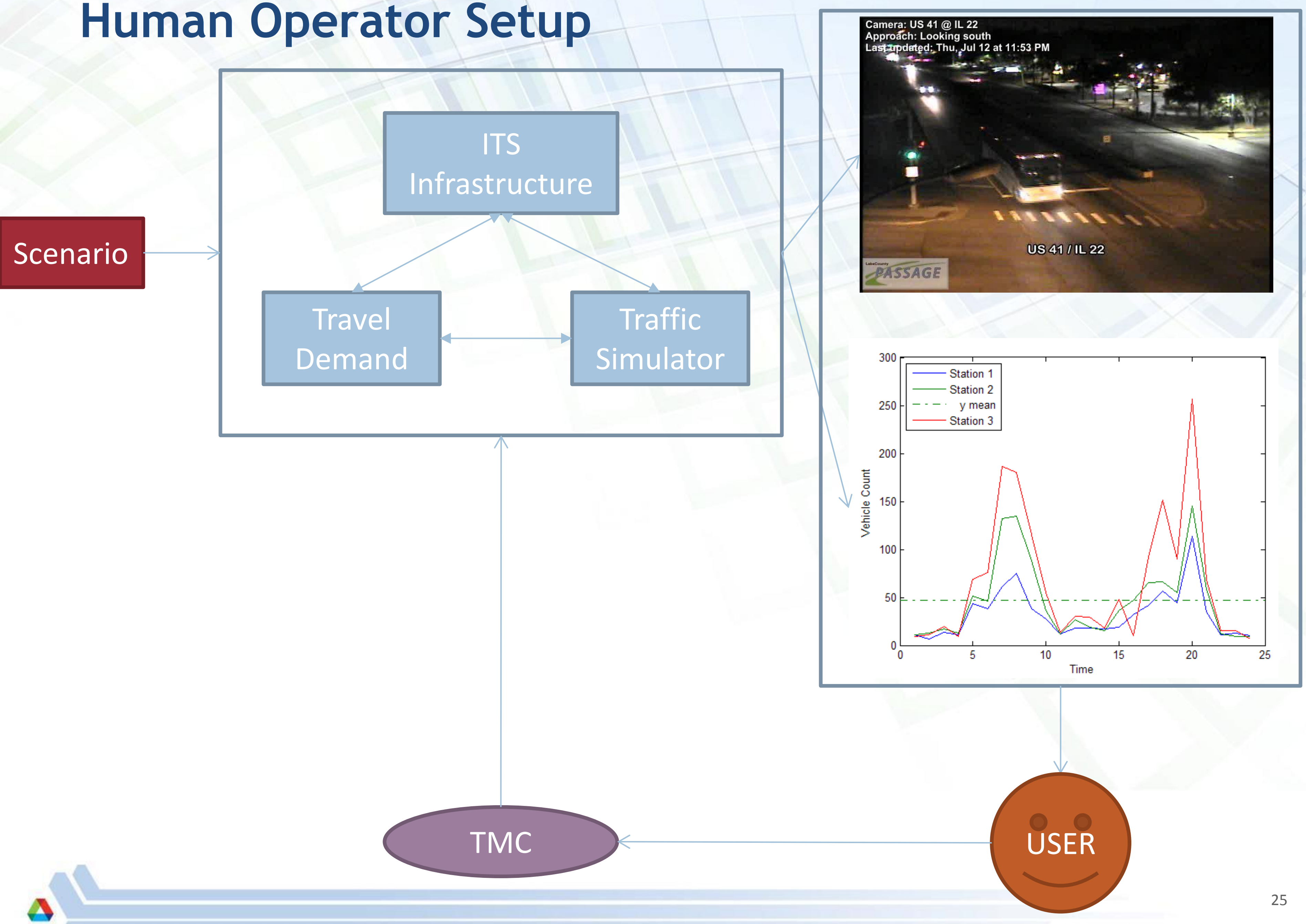
Project Objectives

- A simulation model of a large scale area control and information systems (ITS)
- Model of traffic management center
- Physical infrastructure model (traffic simulation)
- Travel demand model
- Standard hardware/interface protocols for live and constructive simulations

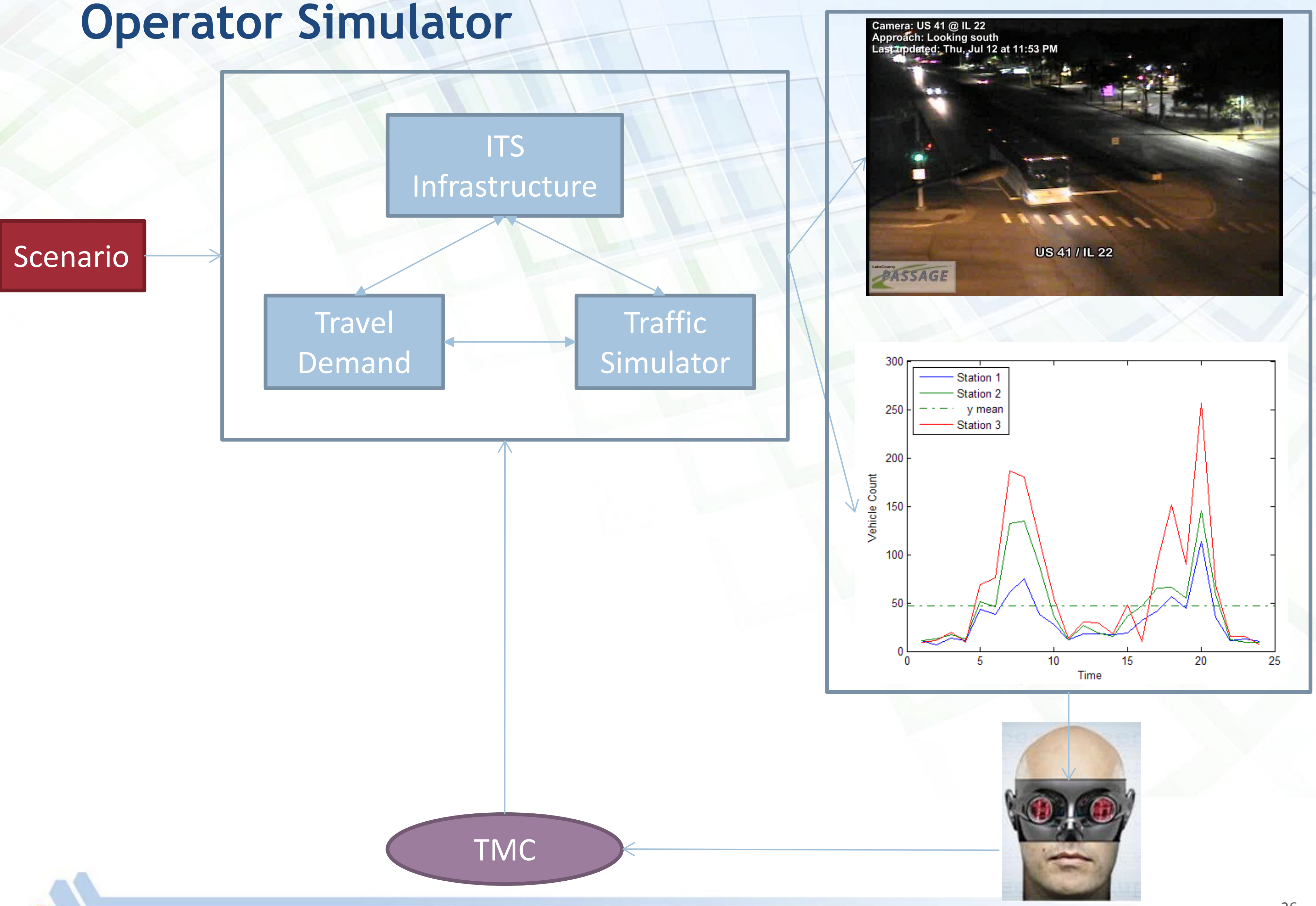
Traffic Analysis Center

- Signals and Controller
  - Fixed Time
  - Actuated
  - Adaptive
- Sensors
  - Loop Detectors
  - GPS probes
  - Video Feeds
  - Toll Stations
- Information Dissemination
  - Enroute travel information
  - Localized pre trip information
  - Broad pre trip travel information
  - Transit
  - Road Network

Human Operator Setup



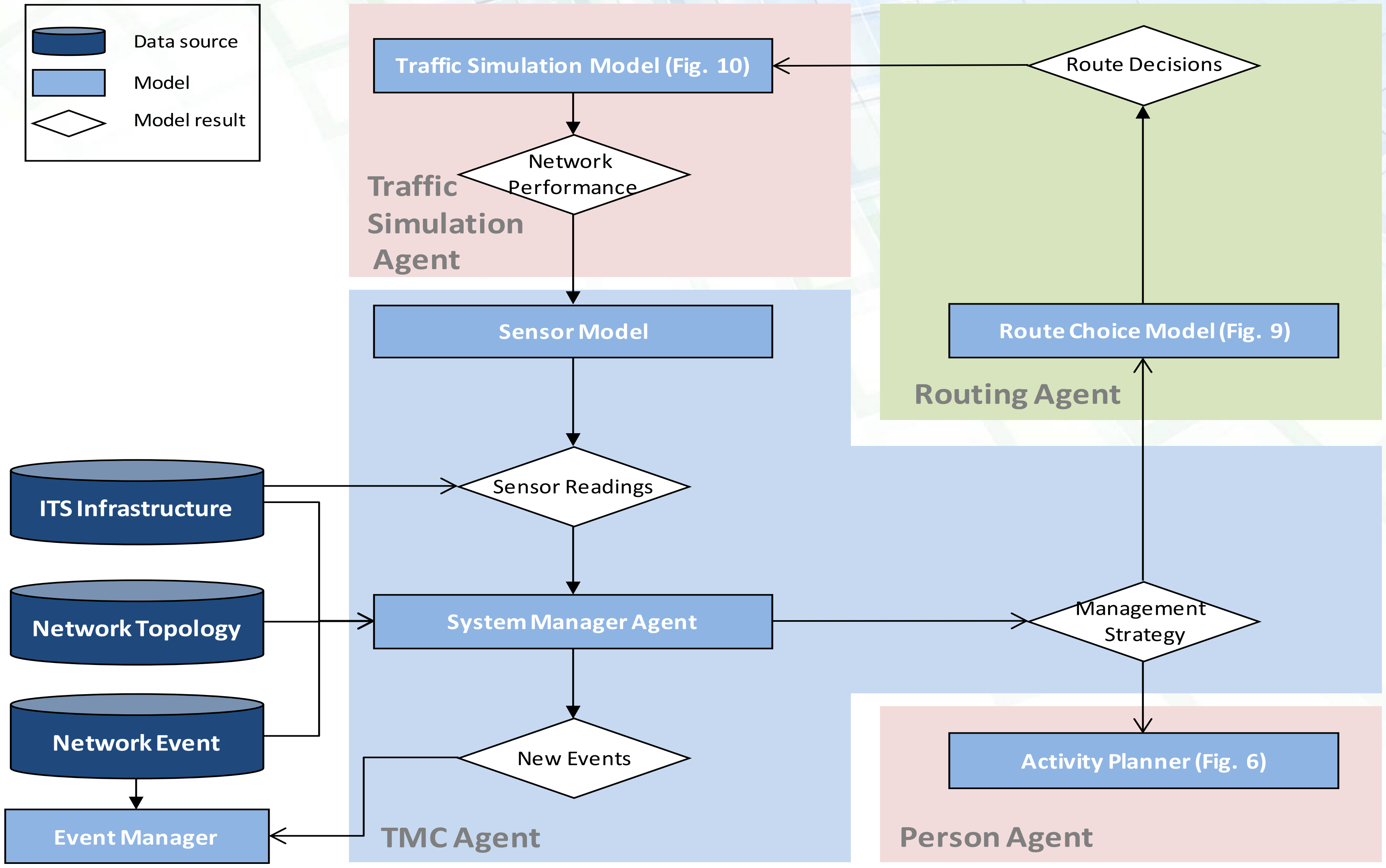
Operator Simulator



Strategies for Prototype

- Real time traffic information system (Web, Radio, GPS, VMS)
  - Travel Times
  - Road construction
  - Traffic accidents
  - Weather
- Real time transit information
  - Normal day schedule
  - Schedule changes
  - Vehicle locations (GPS)
  - Delays
- Managed traffic signals
- Lane management
- Adaptive parking pricing
- Congestion pricing

Agent-Based Intelligent Transportation System Model



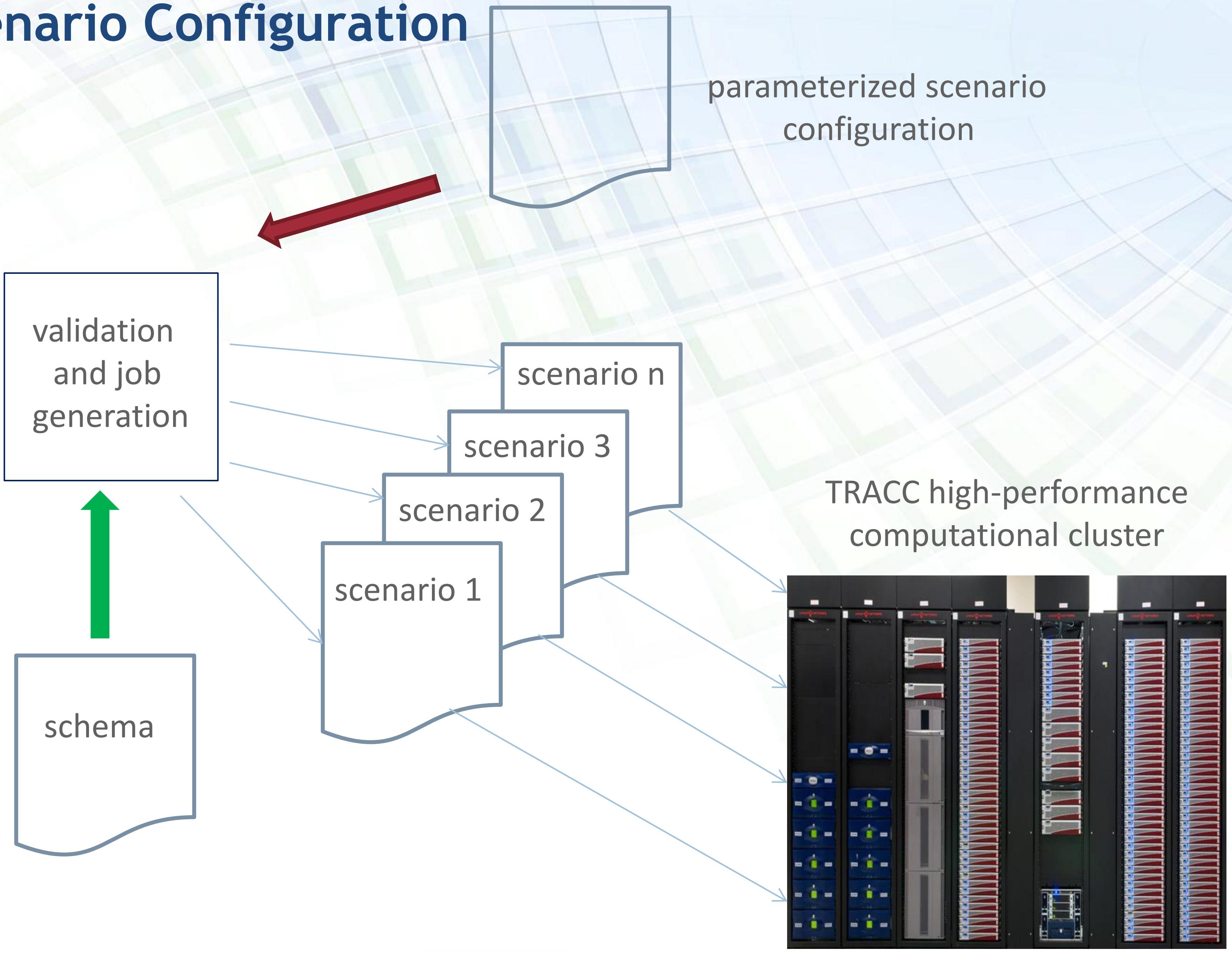
Running Cases

Scenario Configuration

- Scenario configuration
  - Semantic-rich for validation of scenario configuration
  - Human readable
  - Parameterized scenarios: probability distributions

- Parallel execution
  - A separate job is generated for each scenario
  - Submission to cluster
  - Report generation

Scenario Configuration


parameterized scenario configuration
validation and job generation
schema
scenario n
scenario 3
scenario 2
scenario 1
TRACC high-performance computational cluster

Scenario Configuration

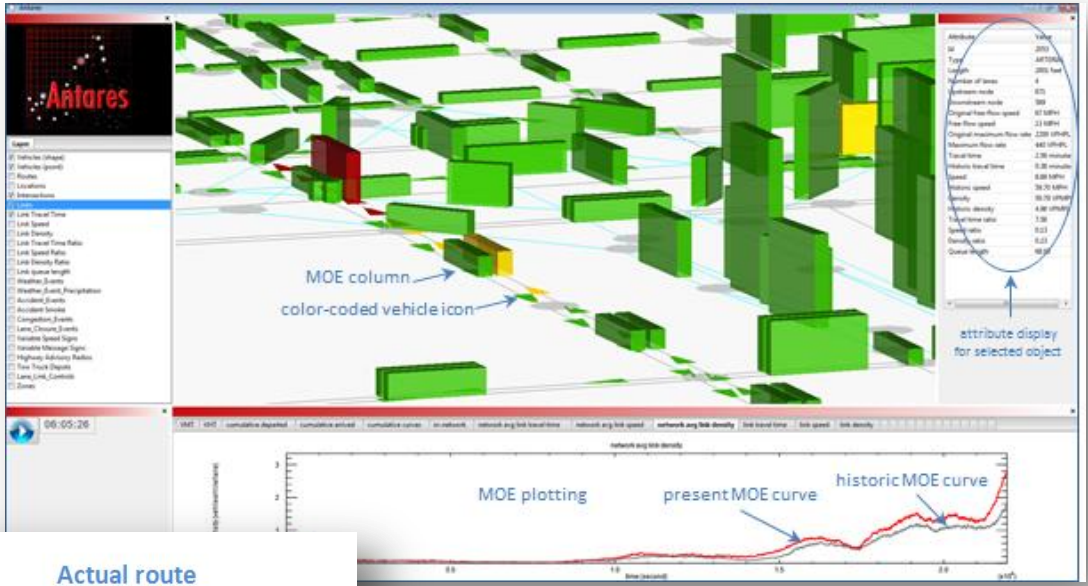
- JSON based for human readability and machine readability
- JSON schema adds power of expressing constraints
  - Type, range
  - Required versus optional
  - Dependency (if one parameter appears another must appear)
  - Exclusion (two parameters cannot appear at the same time)*
  - Functional relationship (one parameter cannot exceed half of another parameter)*
  - Enumeration (one parameter can only take value from a set of values)
  - File existence, privilege*

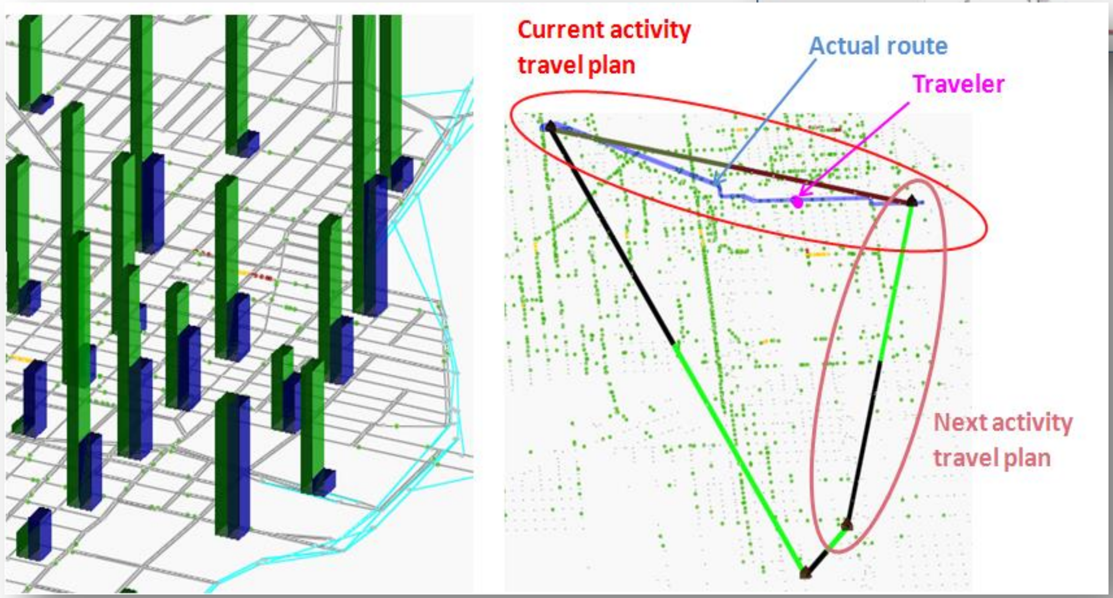
Run Time

- Initial tests were run using the network simulation model to route and simulate 27 million trips in the Chicago metropolitan network over a 24 hour period
- These tests were performed on one node of TRACC's Zephyr cluster, the system specification is as follows: two AMD 6273 2.3GHZ CPUs, 64G RAM, CentOS/Linux 6.2
- The wall time for a multi-threaded case is approximately 75 minutes

Antares Graphical Library

- Graphics Library
- Visualize in 3D or plot in 2D
- WxWidgets
- OpenGL
- plplot
- Multi-threaded

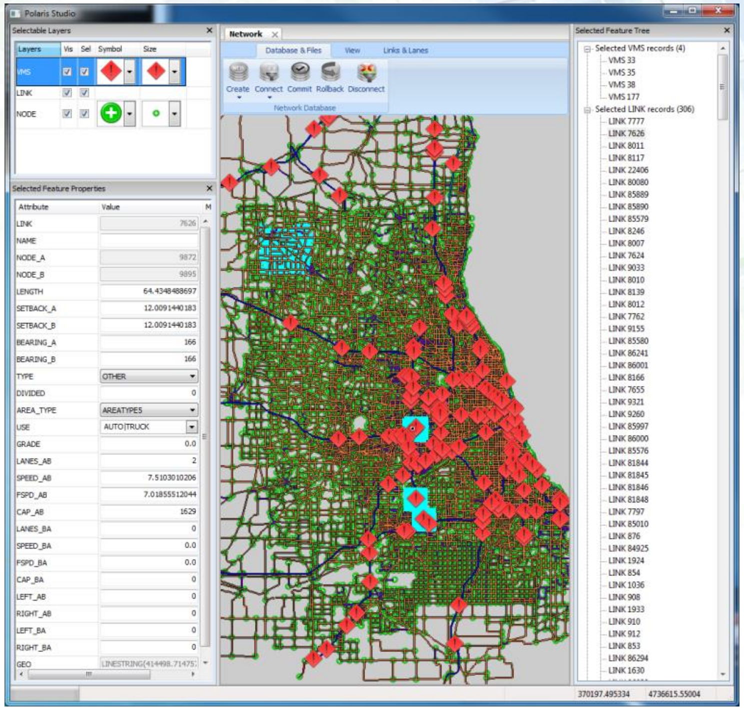




- Organized Into Developer Allocated Layers
- Fully Integrated with Simulation and Core
- Identification capabilities
- Interactivity via identification events

Network Editor

- Intersection Editor
- Layer-Based Drawing
- Label Capabilities
- Selection and Identification Capabilities
- Rule-based Editing



Selected Feature Properties

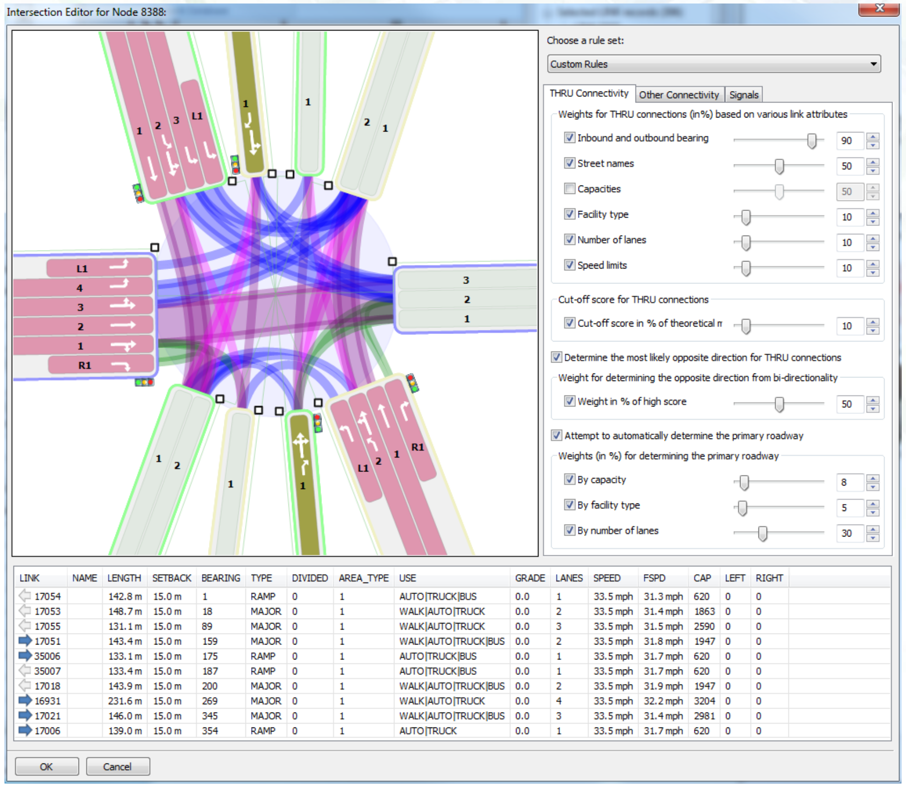
Attribute	Value
LINK	7626
NAME	
NODE_A	9872
NODE_B	9895
LENGTH	64.4348488697
SETBACK_A	12.0091440183
SETBACK_B	12.0091440183
BEARING_A	166
BEARING_B	166
TYPE	OTHER
DIVIDED	0
AREA_TYPE	AREATYPE5
USE	AUTO|TRUCK
GRADE	0.0
LANES_AB	2
SPEED_AB	7.5103010206
FSPD_AB	7.01855512044
CAP_AB	1629
LANES_BA	0
SPEED_BA	0.0
FSPD_BA	0.0
CAP_BA	0
LEFT_AB	0
RIGHT_AB	0
LEFT_BA	0
RIGHT_BA	0
GEO	LINESTRING(414498.714757

Selected VMS records (4)
- VMS 33
- VMS 35
- VMS 38
- VMS 177

Selected LINK records (306)
- LINK 7777
- LINK 7626
- LINK 8011
- LINK 8117
- LINK 22406
- LINK 80080
- LINK 85889
- LINK 85890
- LINK 85579
- LINK 8246
- LINK 8007
- LINK 7624
- LINK 9033
- LINK 8010
- LINK 8139
- LINK 8012
- LINK 7762
- LINK 9155
- LINK 85580
- LINK 86241
- LINK 86001
- LINK 8166
- LINK 7655
- LINK 9321
- LINK 9260
- LINK 85997
- LINK 86000
- LINK 85576
- LINK 81844
- LINK 81845
- LINK 81846
- LINK 81848
- LINK 7797
- LINK 85010
- LINK 876
- LINK 84925
- LINK 1924
- LINK 854
- LINK 1036
- LINK 908
- LINK 1933
- LINK 910
- LINK 912
- LINK 853
- LINK 86294
- LINK 1630

370197.495334 4736615.55004

- General Purpose Network Editor
- WxPython
- OpenGL
- Spatialite
- Sqlite Database



Intersection Editor for Node 8388:

Choose a rule set: Custom Rules

THRU Connectivity | Other Connectivity | Signals

Weights for THRU connections (in%) based on various link attributes
- ☑ Inbound and outbound bearing: 90
- ☑ Street names: 50
- ☐ Capacities: 50
- ☑ Facility type: 10
- ☑ Number of lanes: 10
- ☑ Speed limits: 10

Cut-off score for THRU connections
- ☑ Cut-off score in % of theoretical m: 10

☑ Determine the most likely opposite direction for THRU connections

Weight for determining the opposite direction from bi-directionality
- ☑ Weight in % of high score: 50

☑ Attempt to automatically determine the primary roadway

Weights (in %) for determining the primary roadway
- ☑ By capacity: 8
- ☑ By facility type: 5
- ☑ By number of lanes: 30

LINK	NAME	LENGTH	SETBACK	BEARING	TYPE	DIVIDED	AREA_TYPE	USE	GRADE	LANES	SPEED	FSPD	CAP	LEFT	RIGHT
⇦ 17054		142.8 m	15.0 m	1	RAMP	0	1	AUTO|TRUCK|BUS	0.0	1	33.5 mph	31.3 mph	620	0	0
⇦ 17053		148.7 m	15.0 m	18	MAJOR	0	1	WALK|AUTO|TRUCK	0.0	2	33.5 mph	31.4 mph	1863	0	0
⇦ 17055		131.1 m	15.0 m	89	MAJOR	0	1	WALK|AUTO|TRUCK	0.0	3	33.5 mph	31.5 mph	2590	0	0
➡ 17051		143.4 m	15.0 m	159	MAJOR	0	1	WALK|AUTO|TRUCK|BUS	0.0	2	33.5 mph	31.8 mph	1947	0	0
➡ 35006		133.1 m	15.0 m	175	RAMP	0	1	AUTO|TRUCK|BUS	0.0	1	33.5 mph	31.7 mph	620	0	0
⇦ 35007		133.4 m	15.0 m	187	RAMP	0	1	AUTO|TRUCK|BUS	0.0	1	33.5 mph	31.7 mph	620	0	0
⇦ 17018		143.9 m	15.0 m	200	MAJOR	0	1	WALK|AUTO|TRUCK|BUS	0.0	2	33.5 mph	31.9 mph	1947	0	0
➡ 16931		231.6 m	15.0 m	269	MAJOR	0	1	WALK|AUTO|TRUCK	0.0	4	33.5 mph	32.2 mph	3204	0	0
➡ 17021		146.0 m	15.0 m	345	MAJOR	0	1	WALK|AUTO|TRUCK|BUS	0.0	3	33.5 mph	31.4 mph	2981	0	0
➡ 17006		139.0 m	15.0 m	354	RAMP	0	1	AUTO|TRUCK	0.0	1	33.5 mph	31.7 mph	620	0	0

OK Cancel

POLARIS Framework

Why Develop a Framework for Building Transportation Models?

- Pattern of extremely common objects being re-written, simply to provide slightly different views. Many models differ primarily in level of aggregation.
- Certain areas (namely Intelligent Transportation Systems) have entities which are rapidly changing and cannot be represented adequately in a black-box model.
- Groups who want to add features or change behavior tend to write new models rather than salvage material from existing models due to the difficulty of re-adpating them, this incurs a re-invention of the wheel.
- Many performance and modularity-enhancing capabilities in the realm of advanced computing are being under-utilized.

Who is the POLARIS User Community?

- Transportation Researchers
  - Test and validate theories in an integrated environment quickly and easily
  - Refine and expand the transportation ontology model
- Integrated Transportation Model Developers
  - Weave together disparate model components developed by researchers
  - Bring in new technologies and connect with existing models of interest
- Transportation Modelers
  - Apply models created by the Integrated Model Developers
  - Solve real world problems using POLARIS

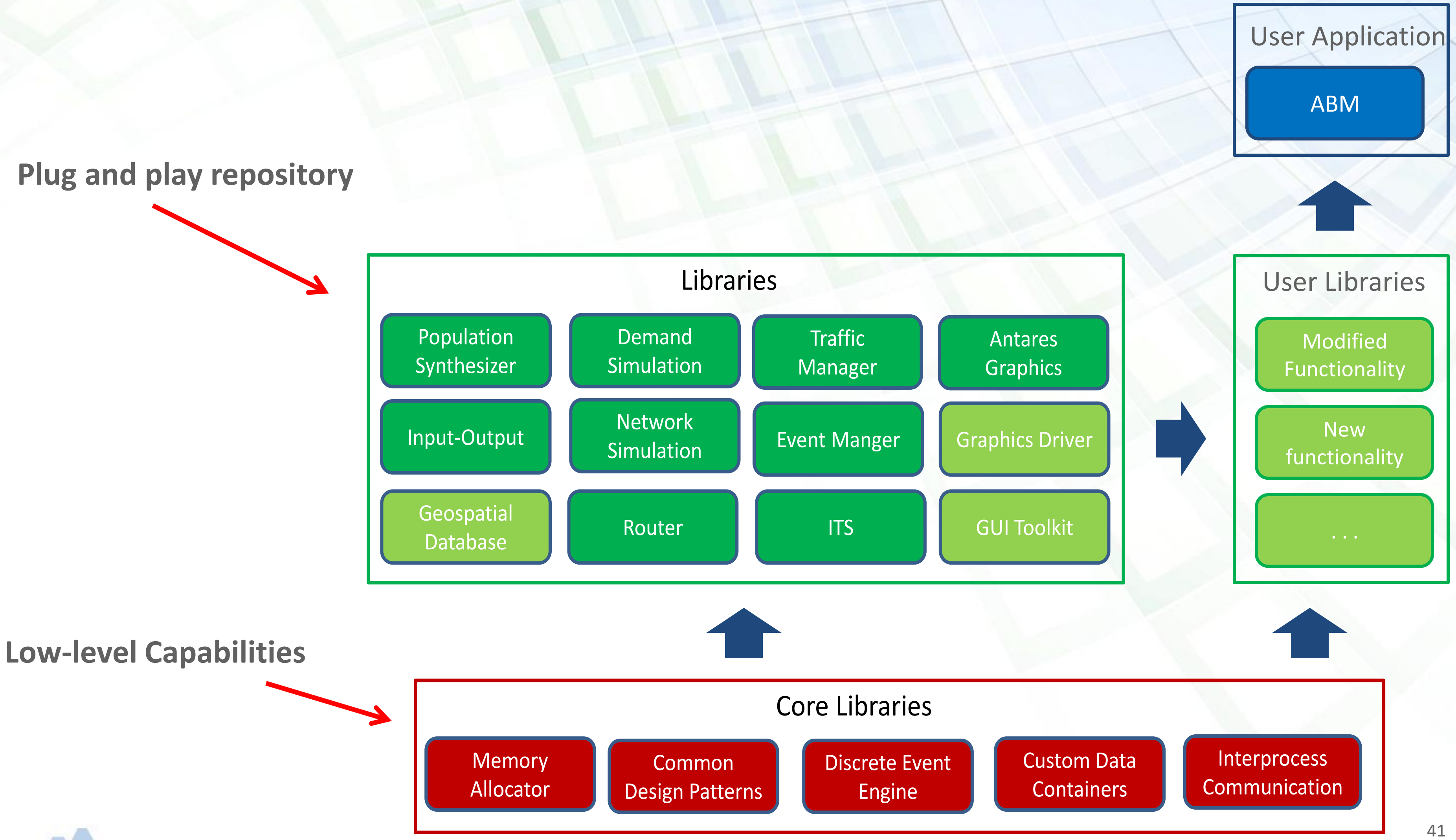
POLARIS Core:
Re-Usable Low Level Capabilities

- Discrete Event Engine
  - Enables writing from an agent-based perspective
- Memory Allocation Library
  - Optimized for the type of allocation needed in transportation modeling applications
- High Performance Data Structures
  - Non-standard structures relevant for use in transportation modeling applications
- Interprocess Engine
  - Enables parallel cluster execution

What Does a Plug and Play Repository Look Like?

- A **collection of model fragments** which **communicate** their capabilities and limitations to other **developers**
- Software written with **common** structural and stylistic elements for **easy** digestion
- **Fundamental** low-level **objects** and **services** which are ubiquitously **re-usable**
- Code **designed** to be easily **adapted** and **revised**
- A **space** for users to develop **new** models **separate** from the **core** repository
- Follows a policy of regularly **integrating** user model **fragments** into a **core** repository which follows a **strict** versioning scheme
- *Developing the structure for this repository is more of a **general** software design challenge whereas the components developed for it are more of a **transportation** software design challenge*

Usage of the POLARIS Modeling Suite



POLARIS Common Design Patterns

- The POLARIS style conventions are generally encapsulated in a series of custom keywords and macros which simplify the development of POLARIS style compliant objects

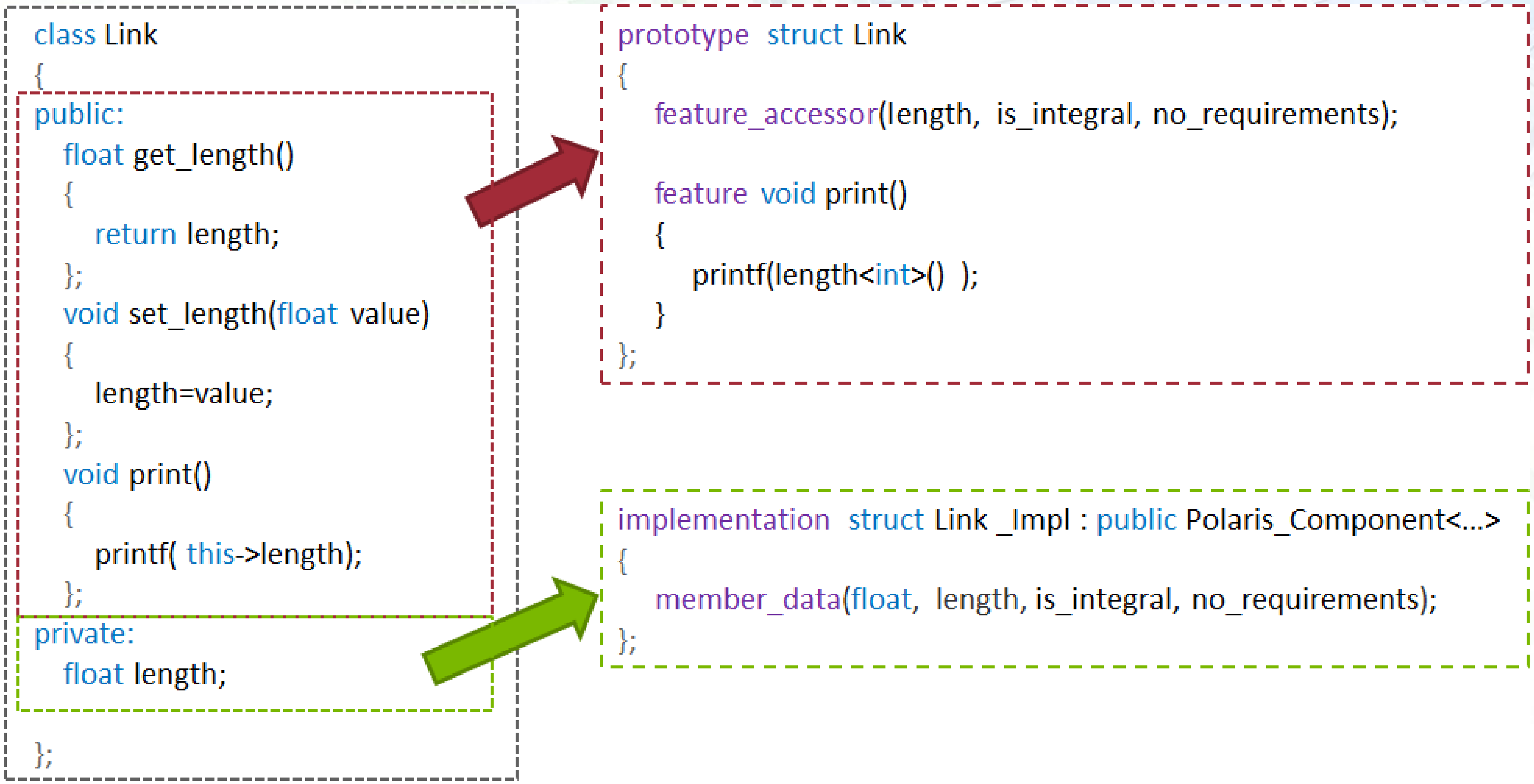
- POLARIS Component: Fundamental POLARIS Type
  - Connects object to memory allocator, interprocess engine, and discrete event engine

- POLARIS Prototype: Extremely Abstract Definition of Type
  - For example: vehicle rather than car, bus, truck

- POLARIS Implementations: Concrete Definition of Type
  - For example: car, bus, truck rather than vehicle

- POLARIS Variables: Basic Types with Relevant Semantic Information
  - Think “feet in meters” instead of “float”

One-to-One Translation of a C++ Class to a POLARIS Component



```
class Link
{
public:
   float get_length()
   {
      return length;
   };
   void set_length(float value)
   {
      length=value;
   };
   void print()
   {
      printf( this->length);
   };
private:
   float length;

};
```

```
prototype  struct Link
{
   feature_accessor(length,  is_integral, no_requirements);

   feature  void print()
   {
       printf(length<int>()  );
   }
};
```

```
implementation  struct Link _Impl : public Polaris_Component<…>
{
   member_data(float,  length, is_integral, no_requirements);
};
```

Thank You