Mega Modeling for Scientific "Big Data" Processing

Stefano Ceri, Emanuele Della Valle
(Politecnico di Milano)
Dino Pedreschi, Roberto Trasarti
(ISTI-CNR and University of Pisa)

The context

Scenario

- BIG DATA: A new data revolution.
- Data is reshaping every individual and collective activity of people's life.
  - Sensors and people produce huge amounts of data
  - Data is becoming accessible everywhere via the Web
- Scientific big data is changing our attitude towards science, from specialized to massive experiments and from focused to broad questions.
- A data-centric vision goes towards Horizon 2020's objectives.

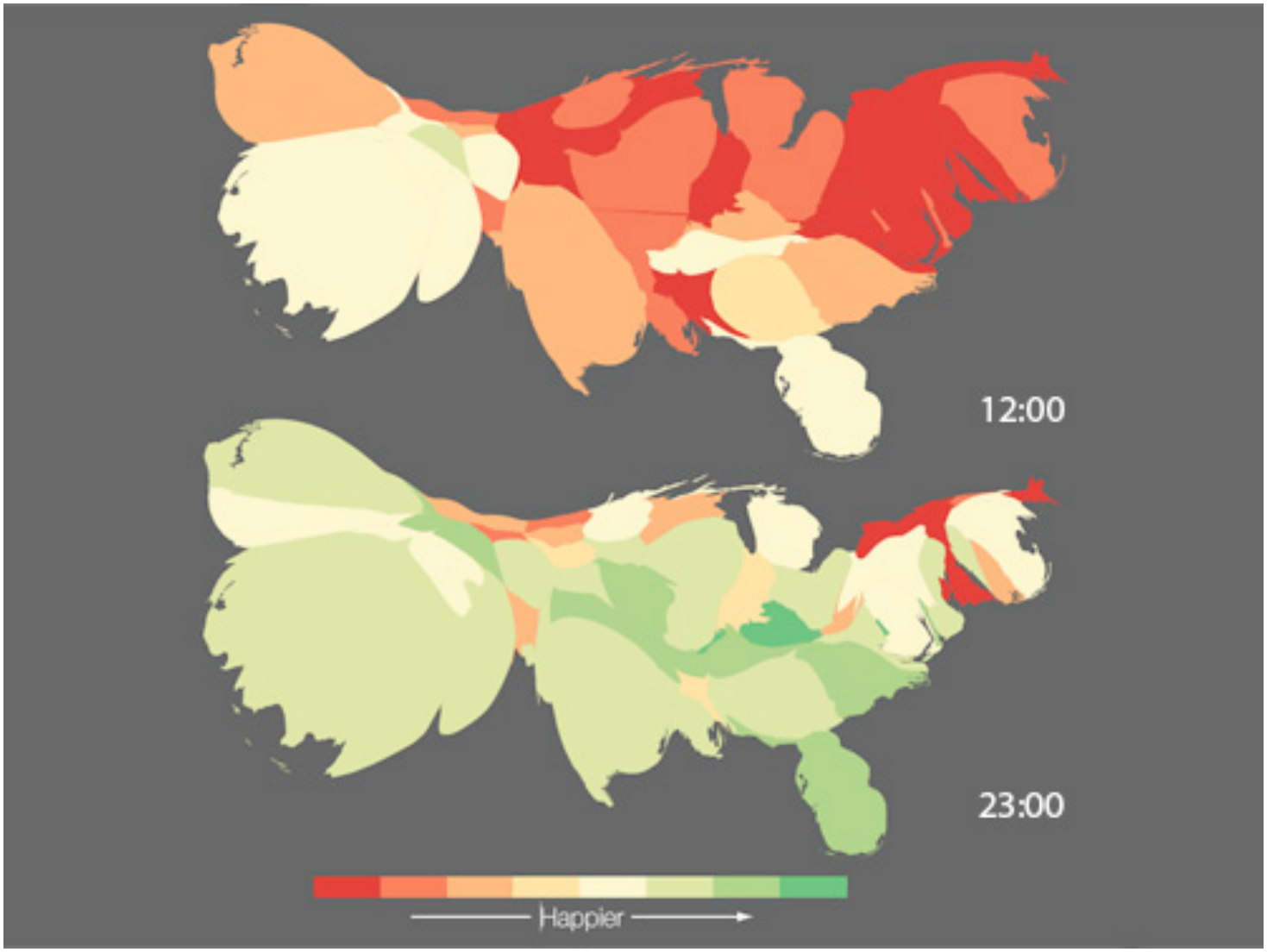
Examples of Big Data
A. London Traffic

Challenges of Scientific Big Data Processing

Smart Cities

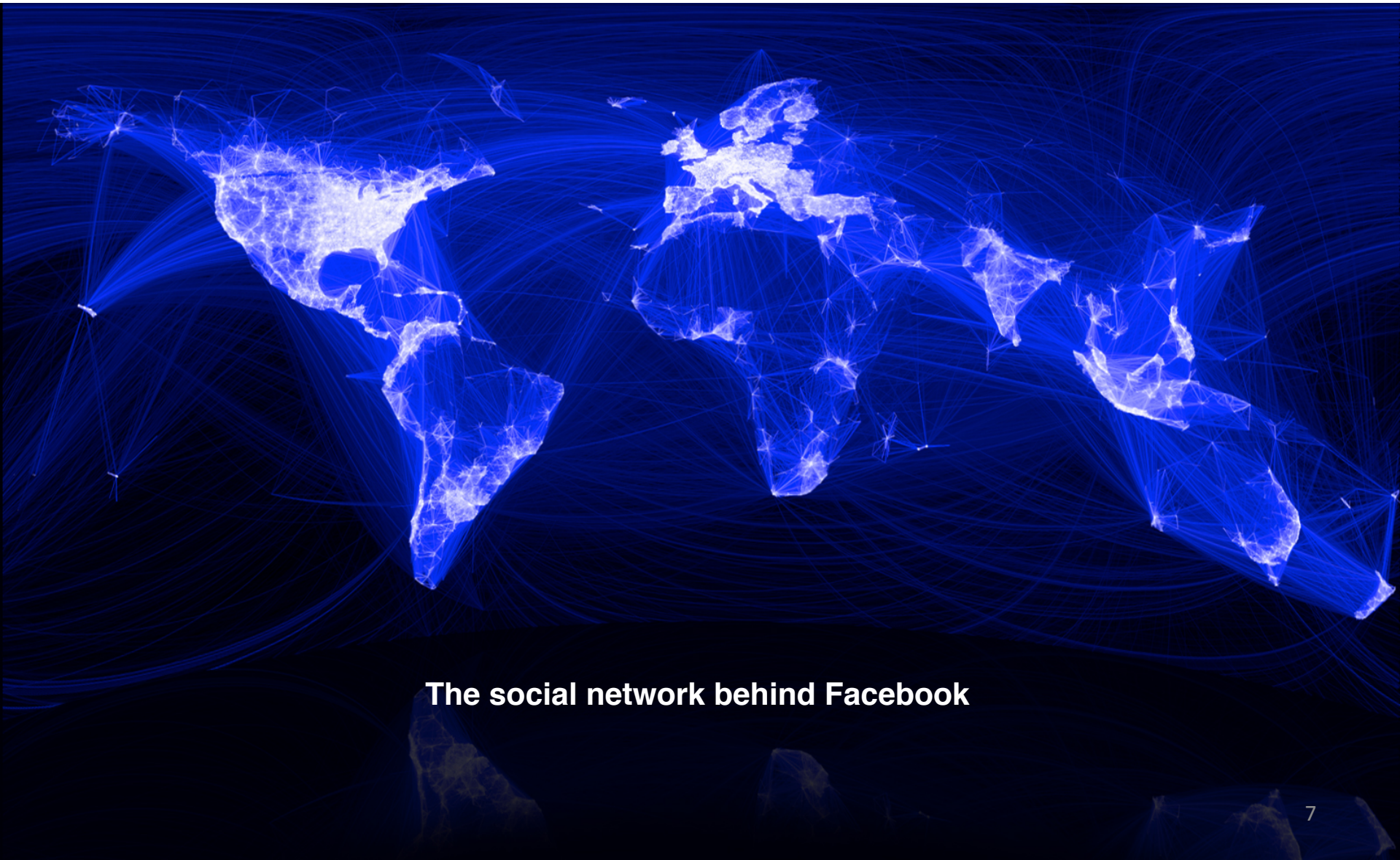
- Cities are becoming smarter, as governments, businesses, and communities increasingly rely on technology to overcome the challenges from rapid urbanization.
- Typical questions for smart cities:
  - Where in the city are people converging during a typical week day? Or during weekends?
  - Is public transportation dynamically adapting to people's density?
  - Is a traffic jam going to happen on this road? And is it then convenient to reallocate travellers based upon the forecast?
  - Where are all my friends meeting? Can I reach them? Should I use public transports or go by car?

B. Pulse of the Nation inferred from Twitter



[source http://www.ccs.neu.edu/home/amislove/twittermood/]

C. Facebook World's Geography

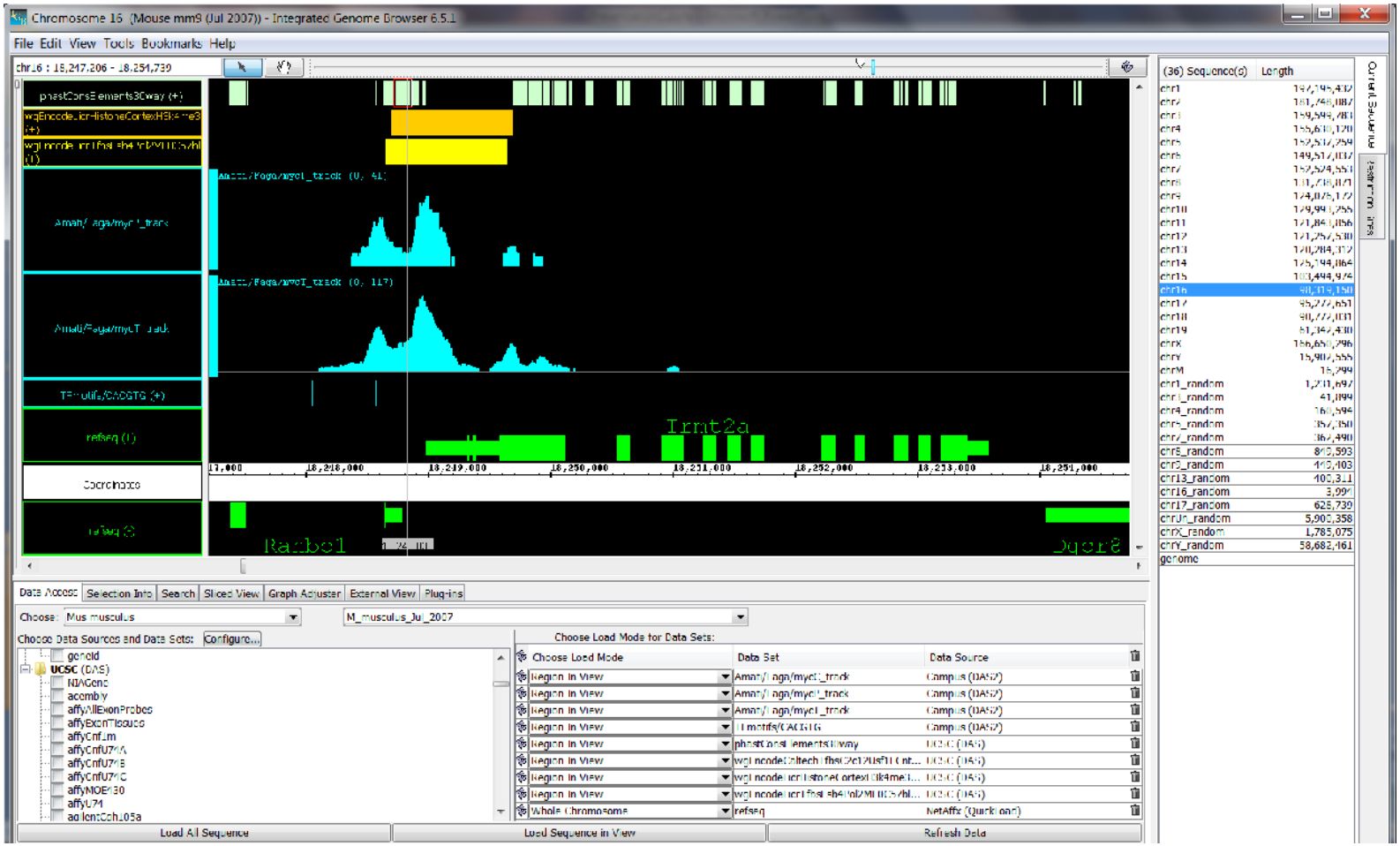


Challenges of Scientific Big Data Processing

Social Mining

- Using user-generated content for discovering and analyzing emergent social behaviors, by combining sensing of personal micro-data (tweets, web logs, mobile phones traces) and participatory sensing (via crowdsourcing, GWAP,…).
- Typical questions for social mining:
  - Who will win US elections? What's the elector's current intention of vote? How reliable is it?
  - Which are the indicators of social well-being (beyond GDP) and how can they be computed and monitored?
  - How is the aging population effectively helped by the social participation to digital community services?
  - What is the link between media ownership and media content? Is there bias in news reporting? And in content reviews?
  - Is an infective disease emerging? How is its diffusion model?

D. Genomic Data



Challenges of Scientific Big Data Processing

Genomic Computing

- The context: thanks to Fast DNA Sequencing, “personalized genomic medicine” will become possible:
  - after a blood sample, with a cost below 100$ and within hours or minutes of computing time, have the entire genome of each individual available at a genome browser
- New questions and scenarios:
  - Am I the carrier of genetic mutations? Will I develop cancer?
  - How obesity correlates with breast cancer?
  - Which computational approach can discriminate between "driver" or "passenger" cancer DNA mutations?
  - How can specific target genes be assigned to epigenetically defined regulatory regions?
  - How do epigenetic modifications affect DNA synthesis during the replication of genomes?

All the scenarios require… MODELS

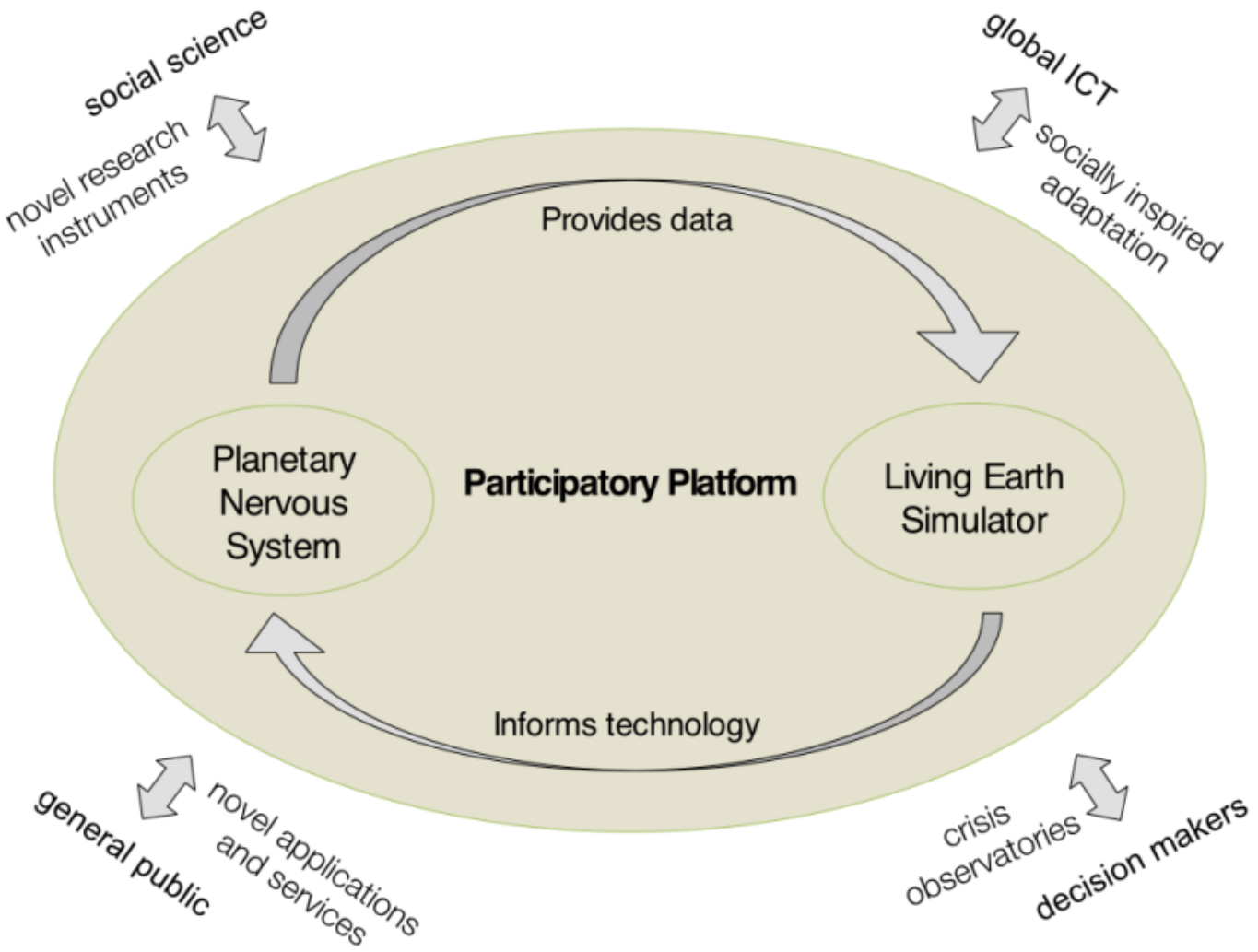
MODEL

- Representation of the problem space in the ICT vocabulary (concepts, data, processes, systems).
- Computational abstractions extracting relevant data from input data
- Models can:
  - Based upon analytical/statistical laws
  - Based upon simulations, extracting general behaviors from many observations of the behavior of individuals
  - Based upon inductive methods applied to data
- Challenge: convergence of three types of models

Motivating Context: FutureICT Flagship

- SCIENCE: The ultimate goal of the FuturICT flagship project is to understand and manage complex, global, socially interactive systems, with a focus on sustainability and resilience.
- POLICY: FuturICT will build a Living Earth Platform, a simulation, visualization and participation platform to support decision-making of policy-makers, business people and citizens.
- TECHNOLOGY: Integrating ICT, Complexity Science and the Social Sciences will create a paradigm shift, facilitating a symbiotic co-evolution of ICT and society.

FuturICT Vision



A stimulus from FuturICT vision: World-of-Modeling Platform

THEORY

- Classify models by type and describe each type's properties.
  - Define **(type-aware) strong interoperability** within the elements of the same class
  - Define **model interoperability** among models of different classes

PRACTICE

- Build language abstractions and software platforms supporting them

Mega-Modeling Concept

Mega-Modeling for Scientific Data

- **General goal**: Building a model of models - which describes each model's properties and interactions - for supporting operations upon models, such as selection, inspection, composition, substitution, reduction, extension, and search.
- **Keywords**: big data, data patterns, management of complexity, uncertainty, dynamic composition, adaptation.
- Chris Welty (Jeopardy): "Increasingly computational tasks require inexact solutions that combine multiple methods in unpredictable ways" (WWW 2012, Lyon)

Which scientific computations?

- **Mathematical model:** uses mathematical concepts and language.
  - **Analytical Model:** mathematical models that have a closed form solution
  - **Numerical Model:** mathematical models that are solved by numerical approximation
- **Statistical model:** uses statistical concepts and language, e.g. probability distribution functions.
  - **Data mining model:** extracts patterns from large data sets.
- **Simulation model:** predicts the expected behavior of a system.
  - **Agent-based model:** simulates the actions and interactions of autonomous agents (representing individuals, groups or organizations)

How should they be modeled?

- By embedding scientific computations within a **conceptual/ontological model** of reality that serves the purpose of defining how computational models share and exchange data, with a clear semantics

The root: Mega-Programming

- Wiederhold-Wegner-Ceri, CACM, Nov. 1992
- Mega-module:
  - Internally homogeneous, independently maintained software system.
  - Each mega-module describes its externally accessible data structures and operations.
- Megaprogramming language MPL
  - A form of programming in the large
- It developed into:
  - “mediators”, “web services”, “Workflow / business process languages”, “semantic web services”, “web 3.0”

Useful ideas of mega-programming

- Every mega-module exposes a data model and certain operations to a mega-program:
  - SUPPLY: provide data in model-compatible format
  - INVOKE: activate computation through entry points
  - EXTRACT: provides mega-module results
  - EXAMINE: makes access to internal state variables
  - ESTIMATE: gets information about execution completion
  - LIMIT: constraints execution time & cost

Previous Uses of Mega-Modeling Term

- BEZEVIN-VALDURIEZ: “On the need for megamodels” (2004), emphasis on meta-models and model registry.
- BEZIVIN: “Model of models” (2004), a model of relationships between models.
- FAVRE: “Meta-model of model transformations” (2005), models linked by relationships such as *representationOf, conformsTo, isTransformedIn.*
- SEIBEL et al. (2010) “dynamic hierarchical data models for traceability” – emphasis on dependencies between model artifacts.
- SEIBEL et al. (2011) mega-models for “modeling runtime behavior”

Data-driven computation paradigms

- *Data analysis*:
  - process of extracting useful information from input data by using any kind of model (including data mining).
- *Data mining:*
  - automatic or semi-automatic analysis of large data sets to *extract previously unknown interesting* **patterns** (emphasis on induction).

On the meaning of pattern

- **Pattern type** = context-independent data format for expressing the results of data analysis and data mining activities – e.g. trajectories
- **Pattern instance** = context-specific data item compliant to the pattern type - e.g. my trajectory from office to home today
- **Pattern** = context-specific population of pattern instances, featuring an intensional description (name, pattern type, qualifying parameters, including quality parameters) and an extension (set of pattern instances) – e.g. the cluster of trajectories leading to Linate airport through the highway
- **Pattern extraction** = computing patterns in a given context, by first evaluating pattern instances and then abstracting the common properties that collectively describe a population

The authors' history of patterns

MineRule Operator (association rules)

- Data type
  - Tabular representation of association rules (HEAD, BODY, SUPPORT, CONFIDENCE)
- Pattern type
  - Association rule HEAD -> BODY, featuring statistical properties of confidence, support
- Paradigm
  - Mine Rule Operator: SQL-based language for extracting association rules and putting them into a tabular format, with built-in variables HEAD, BODY, SUPPORT, CONFIDENCE

Mine Rule Pattern

MINE RULE PurchaseBasket AS

SELECT DISTINCT l..n item AS BODY, I..1 item AS HEAD, SUPPORT, CONFIDENCE

FROM Purchase

WHERE DATE BETWEEN 1-1-2011 AND 1-1-2012

GROUP BY Transaction

HAVING COUNT(*) >= 3

EXTRACTING RULES WITH SUPPORT: 0.2, CONFIDENCE: 0.2

Associations

body	head	support	confidence
ski_pants	jacket	0.2	0.25
hiking_boots	jacket	0.25	0.3
ski_pants, hiking_boots	jacket	0.5	0.3
col_shirt	jacket	0.3	0.2
col_shirt ,hiking_boots	jacket	0.5	0.2

Stream Reasoning

- Data Types
  - RDF Stream: unbound sequence of timestamped RDF triples
  - Window (sliding or tumbling): top portion of the RDF stream
  - Time stamp function: associated to triples
- Pattern Type
  - Computation of a new stream from data and streams
- Paradigm
  - Addition to standard Sparql of new data types and of continuous semantics (i.e., streams and registered queries over streams)

An Example of C-SPARQL Stream

Who are the opinion makers? i.e., the users who are likely to influence the behaviour of other users who follow them

```
REGISTER STREAM OpinionMakers COMPUTED EVERY 5m AS
CONSTRUCT { ?opinionMaker sd:about ?resource }
FROM STREAM <http://streamingsocialdata.org/interactions>
   [RANGE 30m STEP 5m]
WHERE {
        ?opinionMaker ?opinion ?resource.
        ?follower sioc:follows ?opinionMaker.
        ?follower ?opinion ?resource.
        FILTER ( cs:timestamp(?follower) >
                  cs:timestamp(?opinionMaker)
                 && ?opinion != sd:accesses )
}
HAVING ( COUNT(DISTINCT ?follower) > 3 )
```

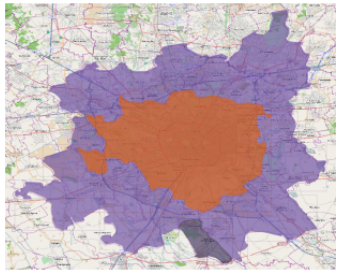
M-Atlas
Interoperability for trajectories

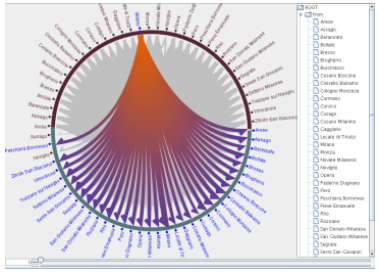
- Data types
  - Points, lines, polygons, trajectories (moving points)
- Patterns
  - Clusters: trajectories of points with the same label
  - Flows: trajectories moving between regions
  - Flocks: spatio-temporal coincidence of flows
- Paradigm
  - SQL-like language for building patterns and for querying, transforming, composing and visualizing them.

M-Atlas queries for social mining

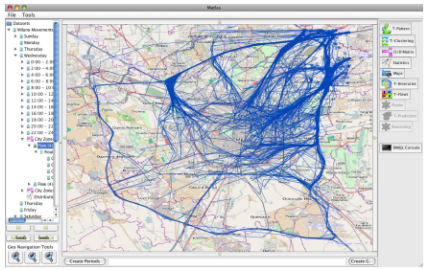
How do people leave Milan's city center toward suburban areas?

```
CREATE MODEL MilanODMatrix AS MINE ODMATRIX
FROM (SELECT t.id, t.trajectory FROM TrajectoryTable t),
(SELECT orig.id, orig.area FROM MunicipalityTable orig),
(SELECT dest.id, dest.area FROM MunicipalityTable dest)
```

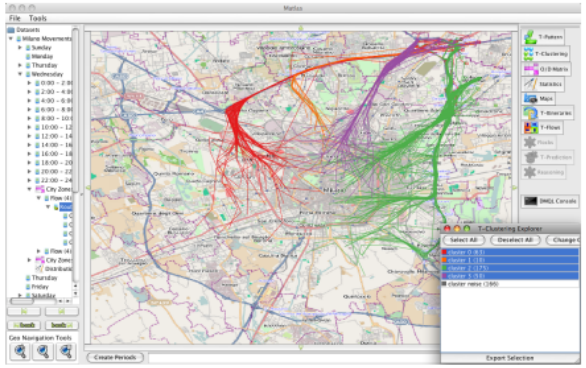


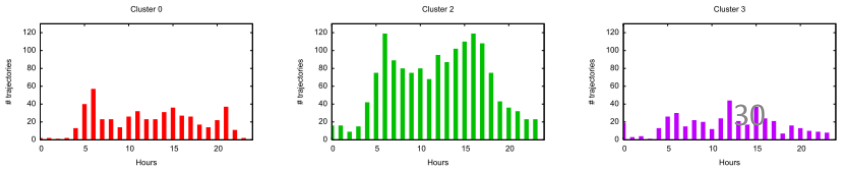


```
CREATE RELATION CenterToNESuburbTrajectories USING ENTAIL
FROM (SELECT t.id, t.trajectory FROM TrajectoryTable t, MilanODMatrix m
WHERE m.origin = Milan AND
m.destination IN (Monza, ..., Brugherio))
```



```
CREATE MODEL ClusteringTable AS MINE T-CLUSTERING
FROM (Select t.id, t.trajectory from CenterToNESuburbTrajectories t)
SET T-CLUSTERING.FUNCTION = ROUTE_SIMILARITY AND
T-CLUSTERING.EPS = 400 AND
T-CLUSTERING.MIN_PTS = 5
```



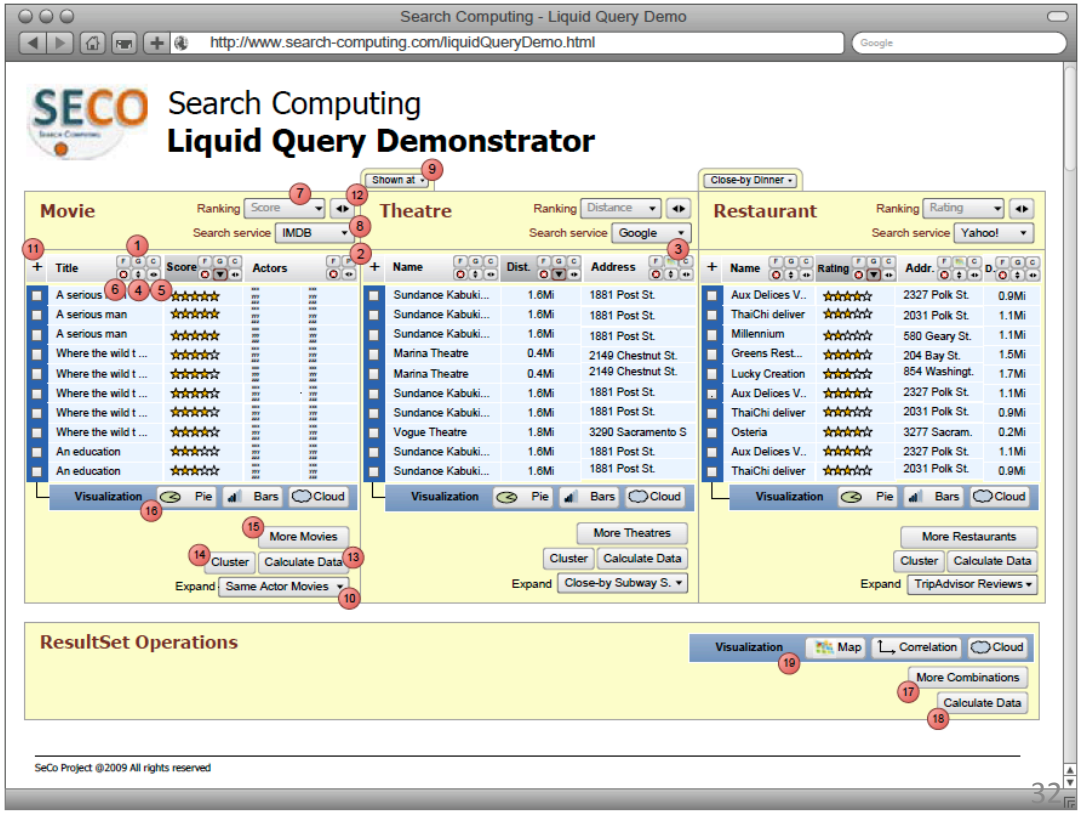


Search Computing

- Data type:
  - Ranked data services with input/output parameters
- Pattern type:
  - Service combinations obtained by computing top-k join queries
- Paradigm:
  - SeCoQL, a query language and protocol supporting ranked queries on services and exploratory search

Search Computing Queries

```
DEFINE QUERY NightPlan($X:String, $Y: string, $Z:Integer , $U:String, $V:String) AS
    SELECT M.*, T.*, R.*, TotalPrice=T.Price + R.AvgPrice
    FROM ((Movie (iGenre: $X, iCountry: Y, iYear: $Z) AS M USING IMDB_MOVIES,
    JOIN Theatre (iAddress: $U, iCity: $V, iCountry: $Y) AS T USING GOOGLE_DISPLAYING ON M.Title=T.Title)
    JOIN Restaurant (iCountry: $Y, iCategory: "Italian Restaurant") AS R USING YQL_LOCAL ON
        T.address=R.Address AND T.city=R.City)
    WHERE R.Rating>3
    RANK BY (R=0.4, T=0.3, M=0.3)
    LIMIT 20 TUPLES AND 50 CALLS
```

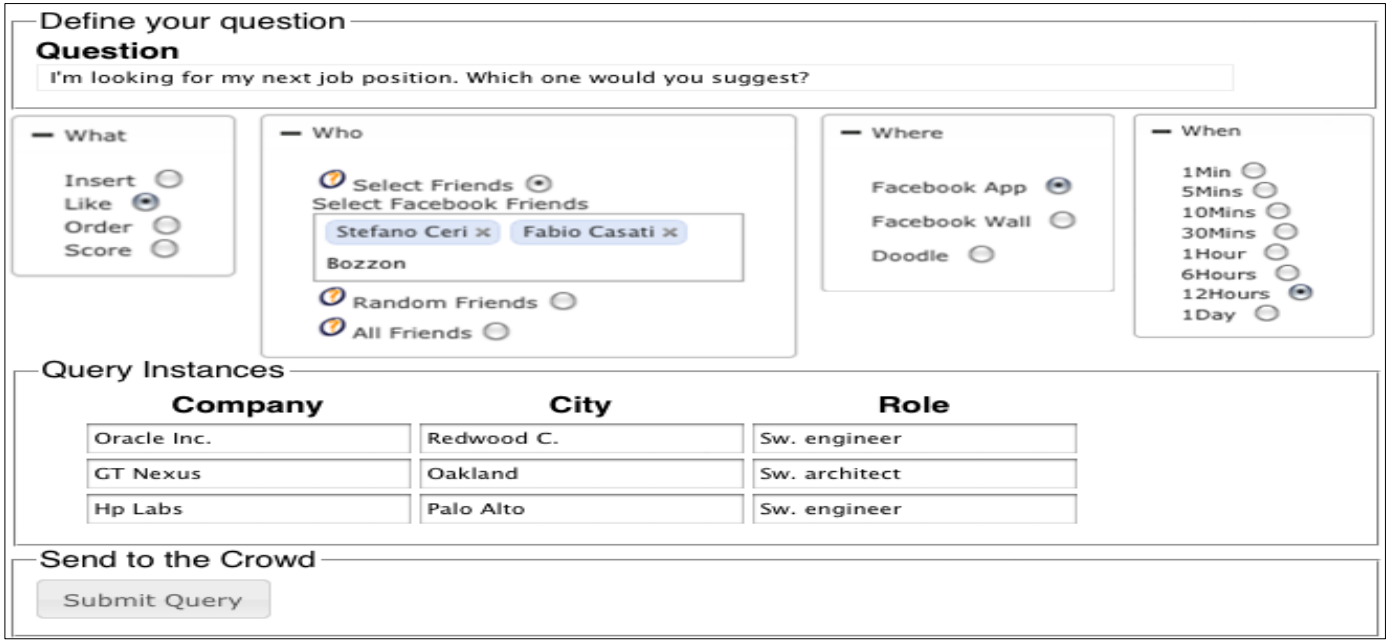


CrowdSearcher

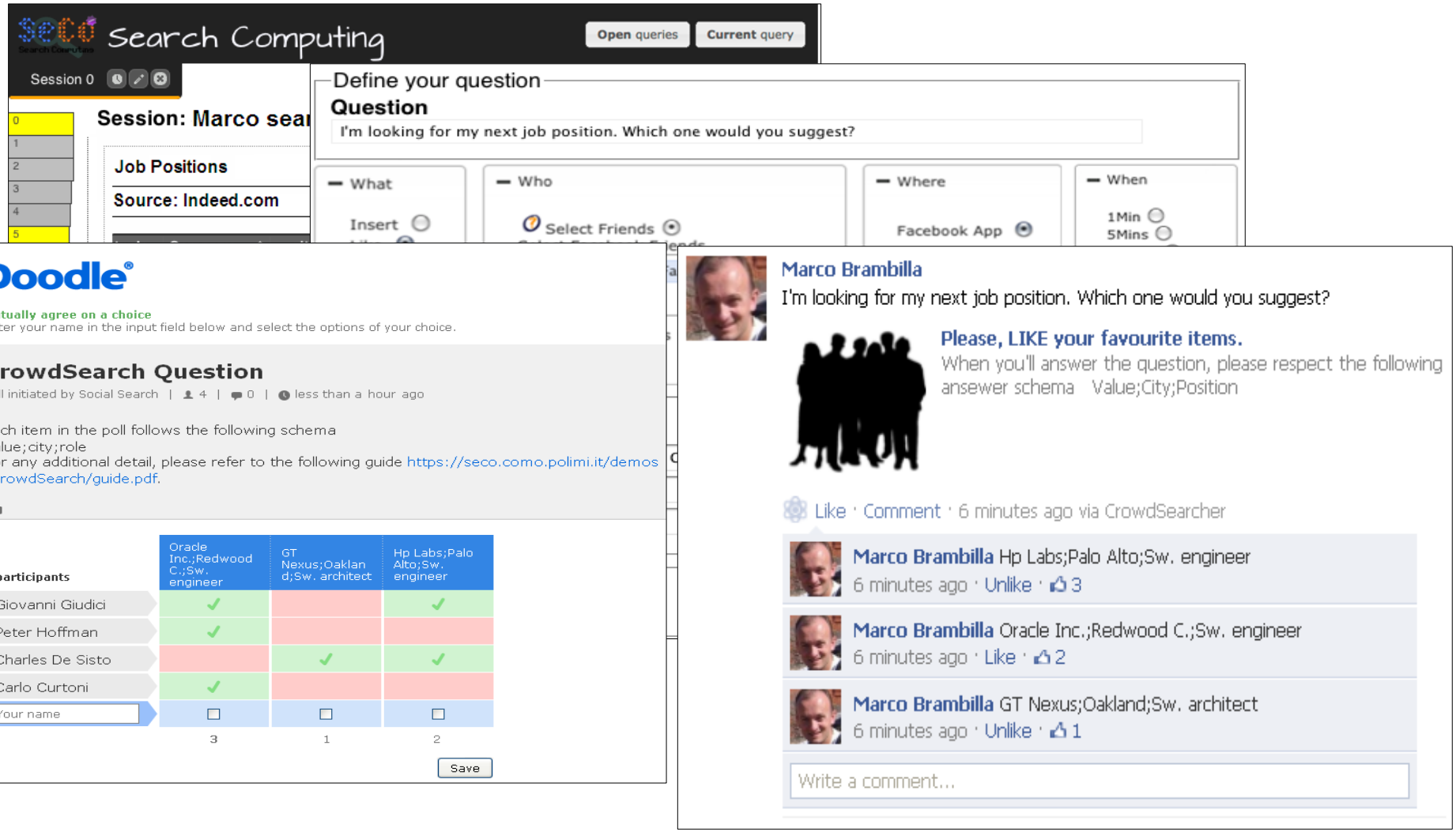
- Data type:
  - List of search items with a regular schema (possibly produced by a conventional search system)
- Pattern types:
  - Annotations on search items (like, dislike, recommend, tag, score, order, group, top, insert delete, correct, connect)
- Paradigm:
  - Use of crowd for adding patterns to search items

CrowdSearcher Model

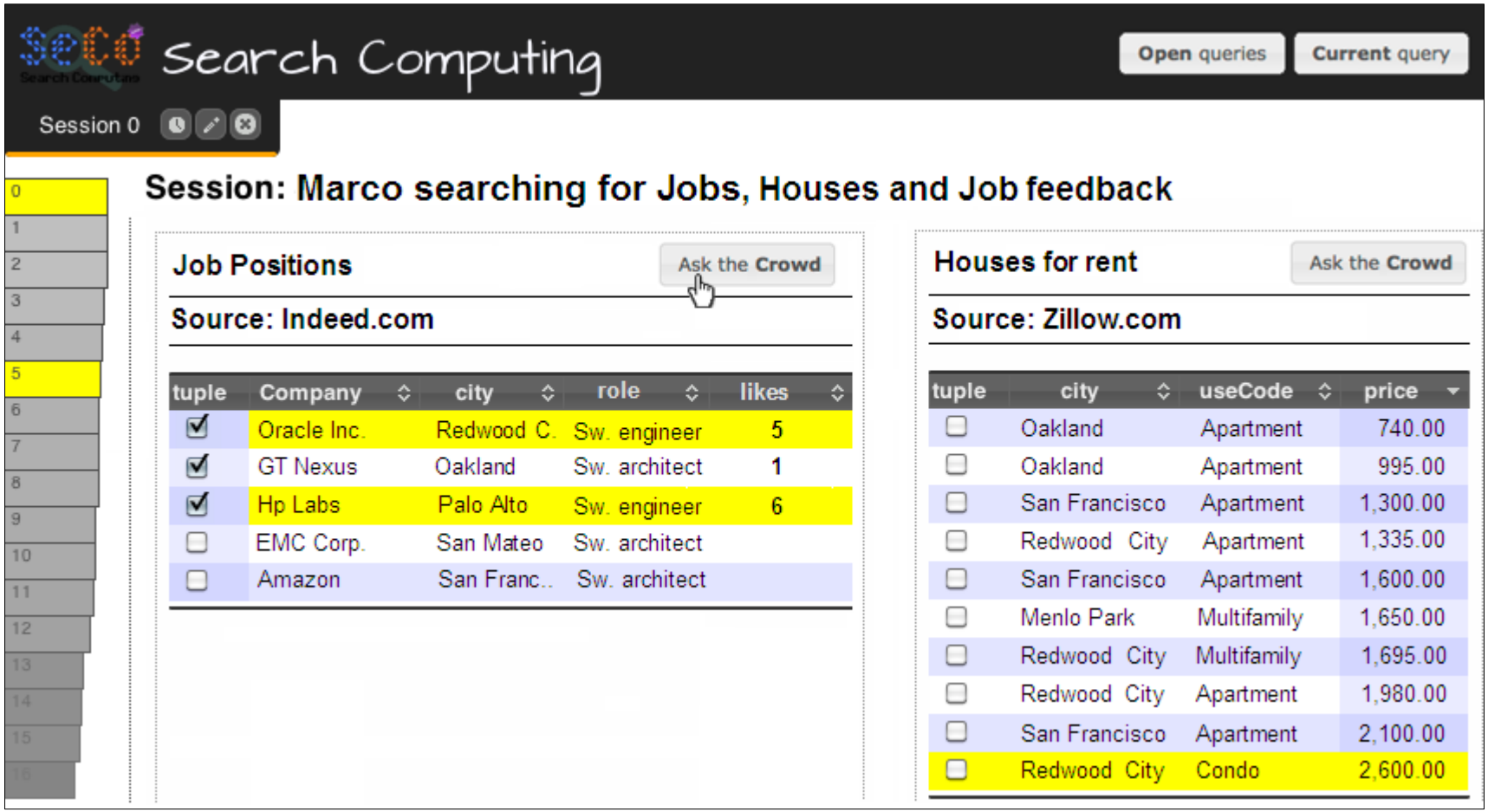
- Data type: collection of tuples
- Query type: Like, Add, Sort / Rank, Comment, Modify



Example of crowdsourcing



Crowdsearcing results



Common aspects of five patterns

- High-level data representation through "tables"
- High-level data manipulation language as an extension of major relational languages, one of: SQL, Sparql, Datalog+-
- Recipe:
  - Expose a tabular representation
  - Use a relational language extension for computation & composition

(just a bit more) Systematic view

Patterns for classification & clustering

- **CLASSIFICATION**. The computation extracts classes from a population, each class has a name and statistics – from simple frequencies up.

  Data: Population(Item)

  Pattern: Class(Name, AggrStats)

- **CLUSTERING**. The computation extracts clusters from a collection, each cluster has a name, an extent (consisting of its elements), a centroid element, and statistics – from cardinalities up.

  Data: Collection(Item)

  Pattern: Cluster(Name, Extent: [Item],
  CentroidItem, AggrStats)

Patterns for Streams

- **STREAMING.** Stream computing aggregates data of a given type from a stream; it associates each type with a valid time interval, typically the most recent, and aggregate properties.

  Data: Stream(TimeStamp, Item)

  Pattern: StreamStats(ItemType, TimeInterval, AggrStats)

- **STREAMING WITH WINDOWS**. The stream is subdivided in windows, stream computing associates a given type and window with aggregate properties.

  Data: Stream(Window, StartTimeStamp,
  EndTimeStamp, Content:[Item])

  Pattern: WindowedStats(Window, ItemType, AggrStats)

Patterns for Association Rules

- **ASSOCIATION RULES.** They solve the basket analysis problem; each association rule has an head and a body describing item sets, and then statistical properties of support and confidence defining the rule's interest.

  Data Basket(Tid,Item)

  Pattern: Rule(Head:[Item], Body:[Item],
  Support, Confidence)

Patterns for Trees

- **TREE.** Classical computations provide the descendants or ancestors of a given node, or classify a new node relative to a taxonomy, by returning the path from the root to the most similar node

  Data: Tree (Item, Children: [Item])

  Pattern: Descendants(Item, To: [Item])

  Ancestors(Item, From: [Item])

  Classify (Item, Path[Item])

Patterns for Graphs

- **GRAPH.** Classical computations provide a decomposition of a graph into components or find the "friend" nodes which are at a given "nearness" from a given node.

  Data: Graph(FromItem, ToItem)

  Pattern: Components(Name, Components: [Node])

  Friends(FromItem, NearnessLevel, To: [Item])

- **DISTANCE-GRAPH.** Shortest path between any two items expressed as a sequence of nodes connecting them and a totaldistance.

  Data: D-Graph(FromItem, ToItem, Distance)

  Pattern: ShortestPath(OriginItem, DestinationItem, Path: [Item], TotalDistance)

Patterns for Moving Points

- **MOVING POINTS**. Reconstruction of the trajectories as sequences of locations which are traversed by the same item.

  Data: Point(Item, Time, Location)

  Pattern: Trajectory(Item, FromLocation, ToLocation,
  Steps:[Location], StepCount: Number)

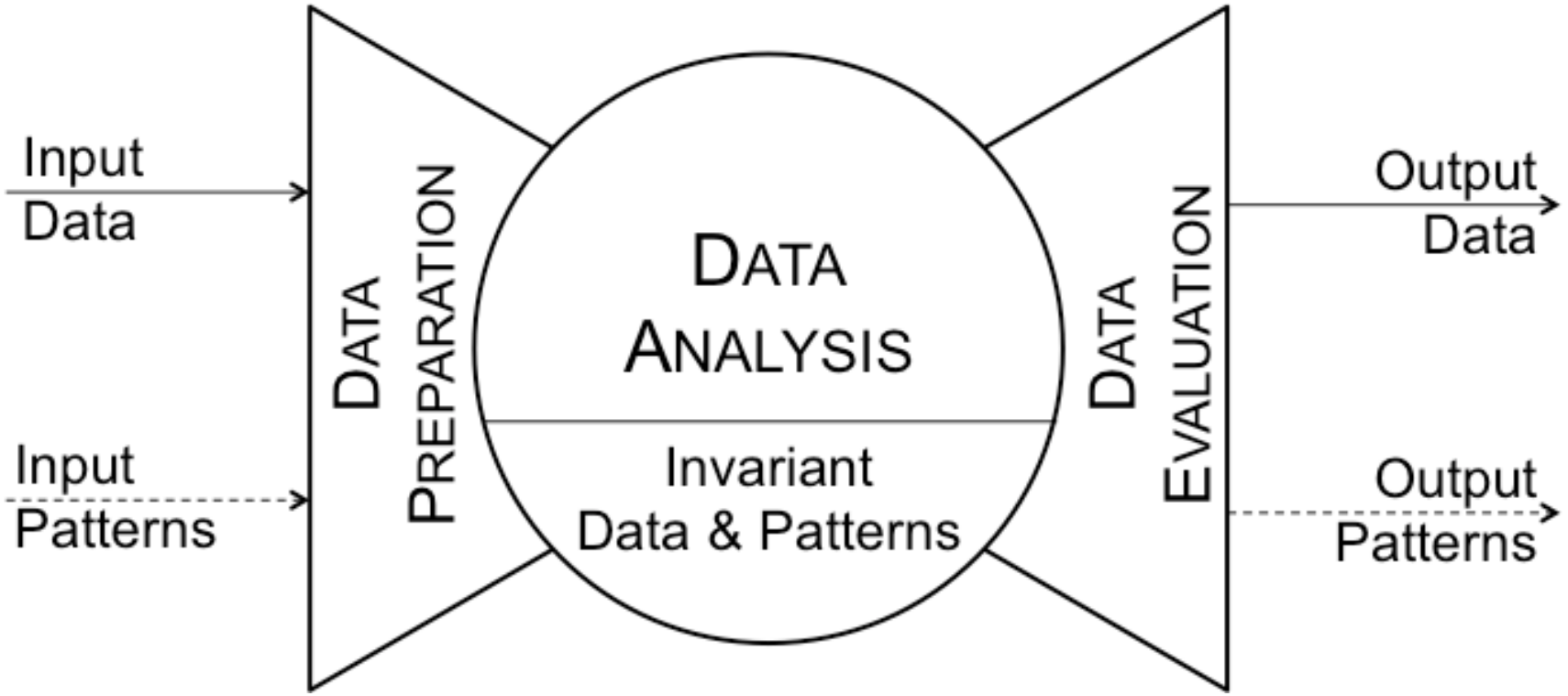
- **FLOCKS**. Combination of trajectories together to recognize flocks, i.e. simultaneous movements of groups of individuals across regions.

  Data: Trajectory(Item, FromLocation, ToLocation,
  Steps:[Location], StepCount: Number)

  Pattern: Flock(FlockName, FromRegion, ToRegion, TimeInterval,
  Objects: [Items], ObjectCount: Number)

(eventually) Mega-modules

Mega-modules



Format

- Data preparation
  - Purpose: assembling input objects --- typically application-specific
  - Techniques: abstraction, semantic enrichment, noise reduction
  - Computation complexity: low (a data scan or sort)
- Data analysis
  - Purpose: performing the core scientific processing, computing output objects --- application-independent
  - Techniques: computational models
  - Computation complexity: as required (partitioning and streaming recommended)
- Data evaluation
  - Purpose: extracting & presenting results --- typically application-specific
  - Techniques: quality assessment, filtering, significance measuring, diversification, ranking
  - Computation complexity: as required (object transformations to fit needs)

Inspections and controls

- Megamodule inspection
  - After preparation: view of input objects
  - After execution: view of output objects
- Megamodule controls
  - Based upon inspection
  - May alter behavior, suspend, resume, terminate

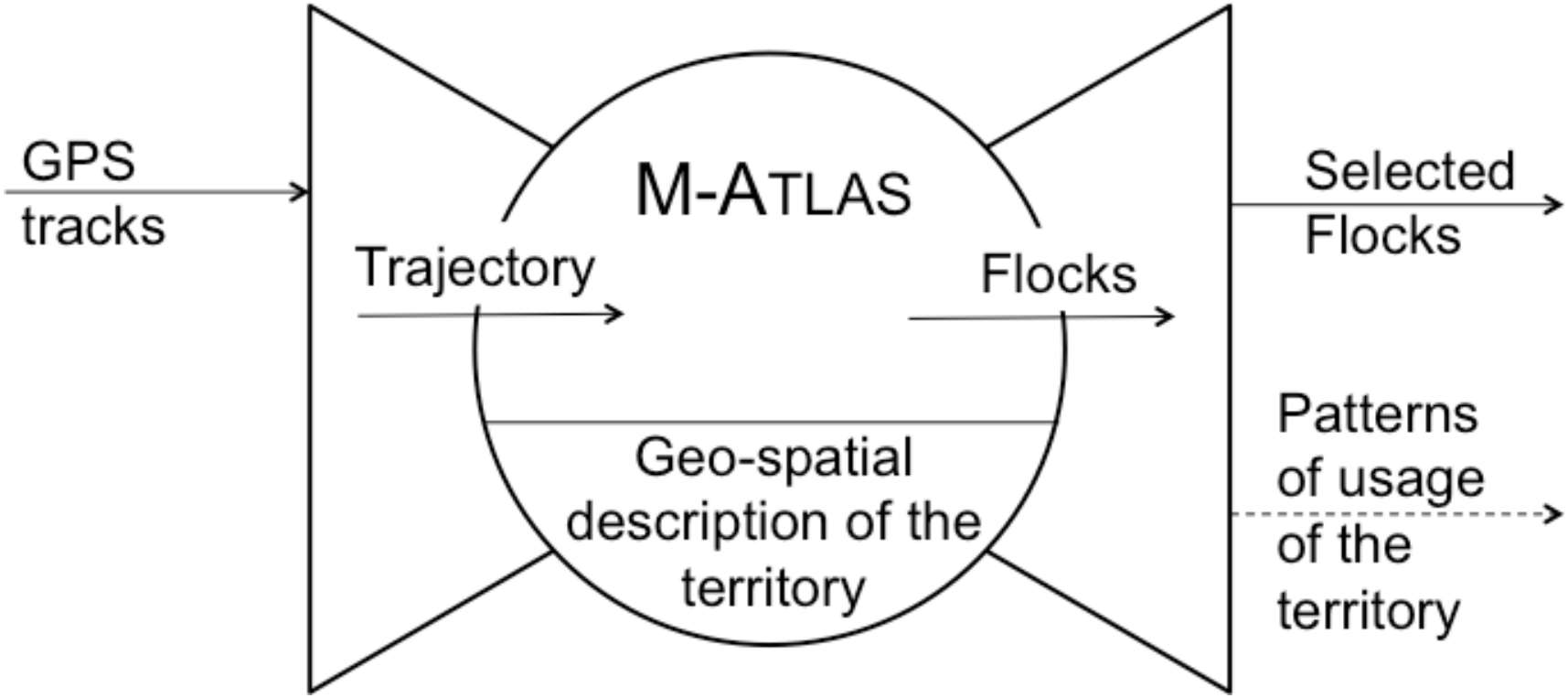
Rationale

- Data analysis: reusable transformation of input objects into output objects
  - Classical mathematical/statistical algorithms *compute* output data
  - Simulation algorithms *predict* output data
  - Data mining methods *induce* output data
- Application-independent input and output objects compliant with pattern types

Relational View of Mega-Modules

- Input/output objects for data analysis in object-relational format?
  - Potential for high-level declarative data analysis description using extended relational query language
  - Easing inspection and control
  - Easing data analysis reuse

Example: M-Atlas



Running Example

- Data preparation
  - GPS observations of the same individual are assembled into a trajectory
- Data analysis
  - Trajectories are assembled and reported as simultaneous movements of groups of people (flocks)
- Data evaluation
  - Flocks which are most relevant (above threshold) are reported upon a map

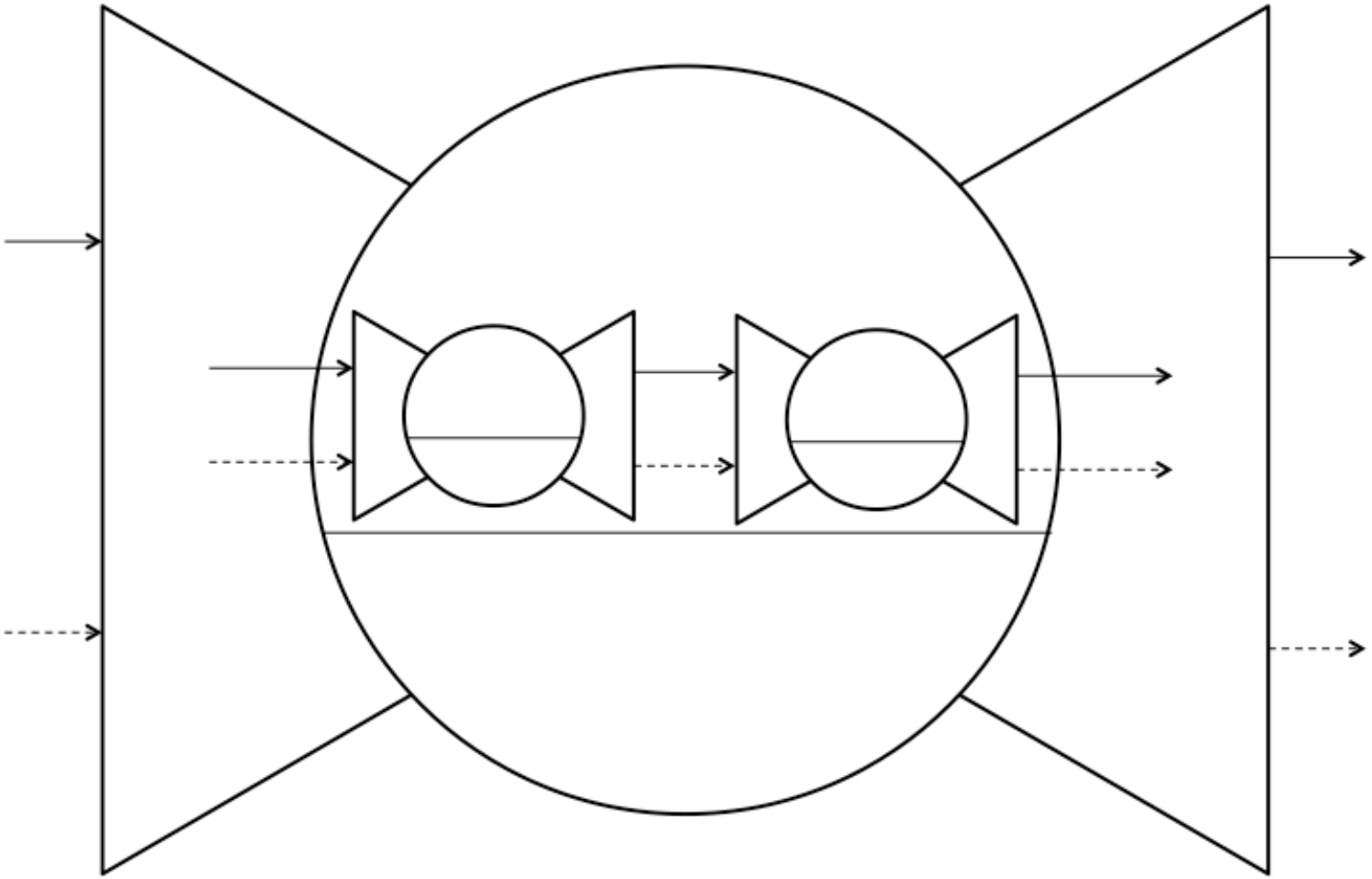
Composition Abstractions

- Used for assembling mega-modules into higher order computations
- If appropriately chosen, are key to mega-module reuse
- Ideal design process = top-down, recursive application of (de)composition abstractions up to finding the appropriate mega-modules within a repository

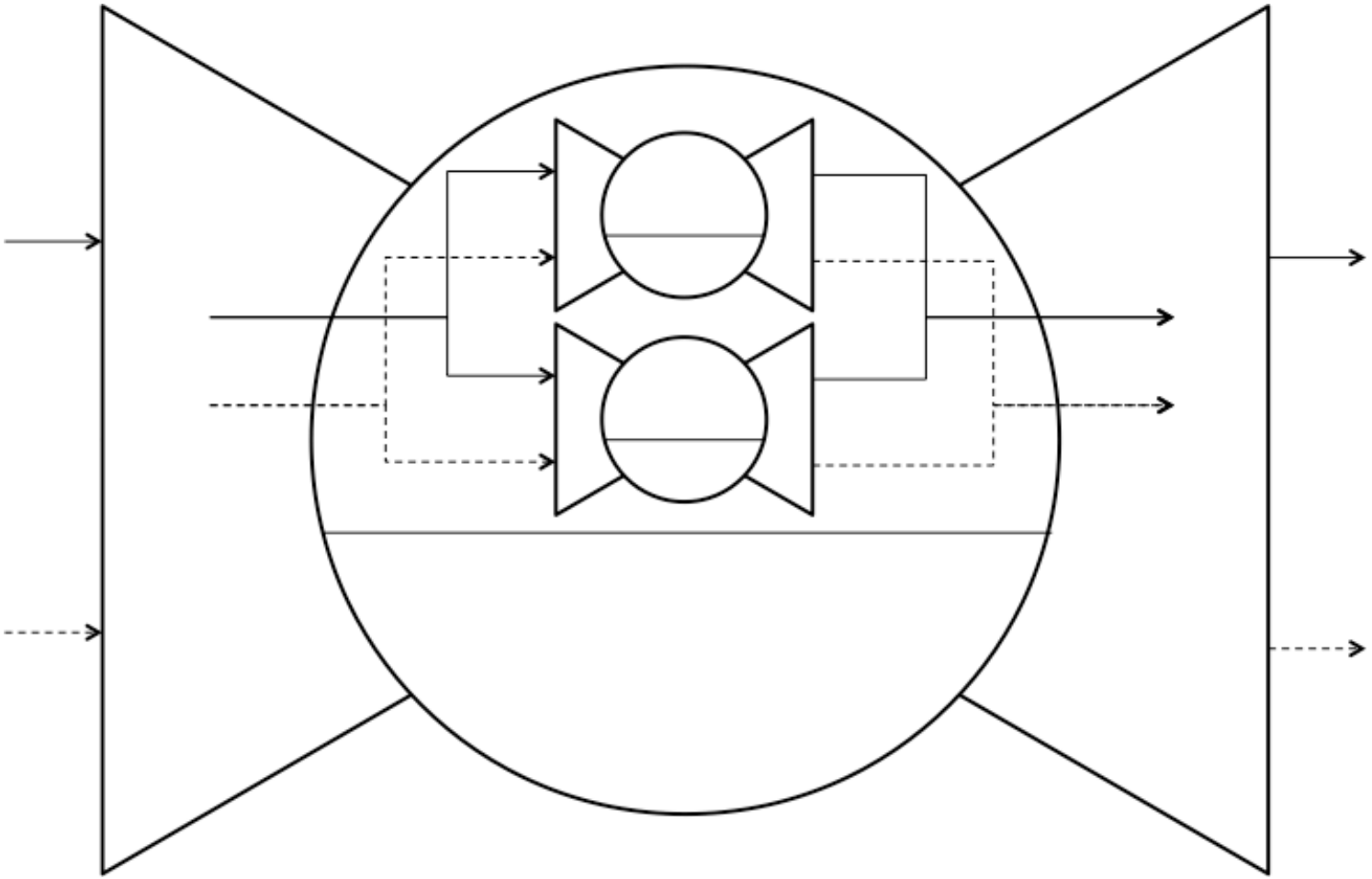
Composition Abstractions (so far)

- General-purpose
  - Pipeline
  - Parallel/Iterative
- Recurrent
  - What-if control
  - Drift control

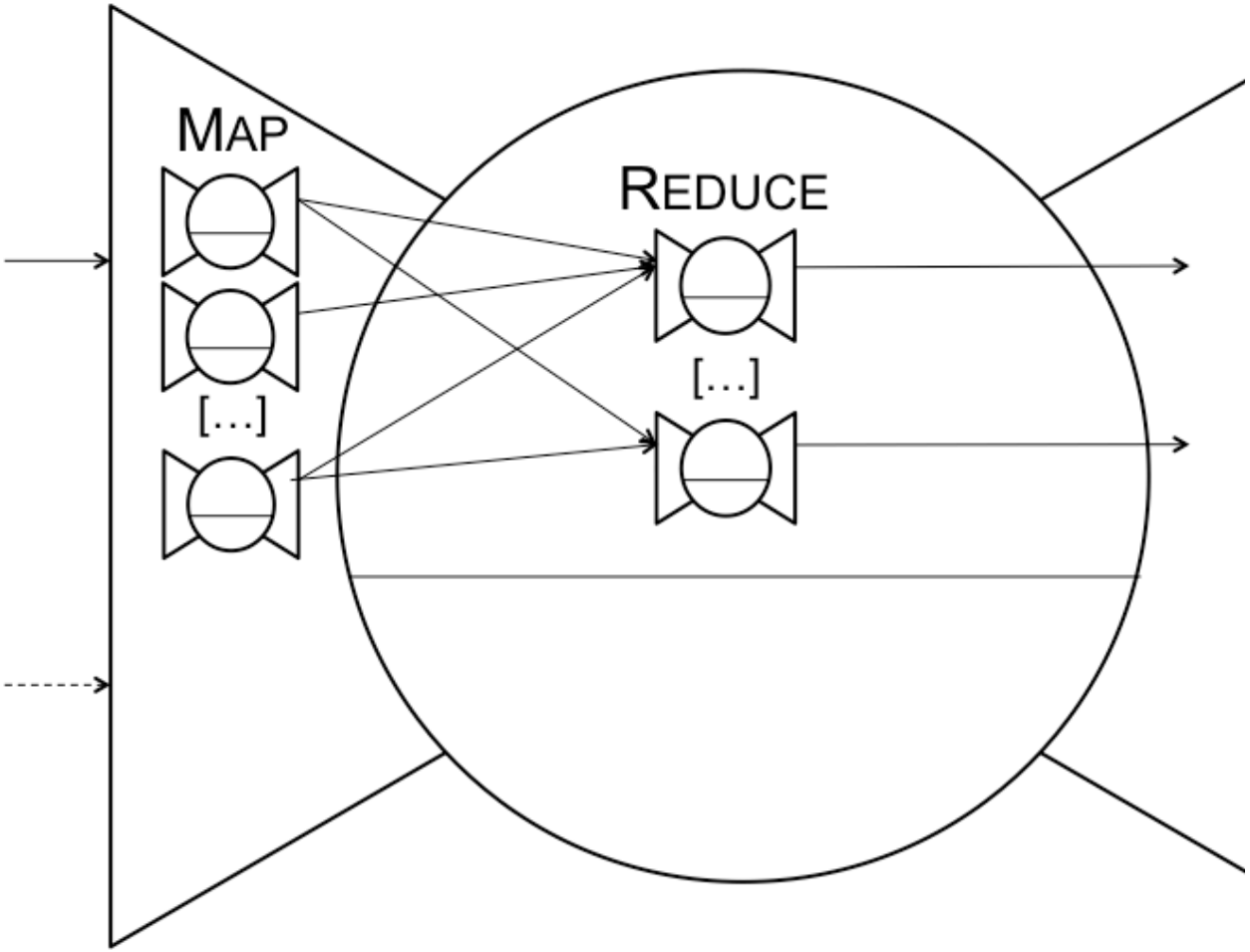
Pipeline



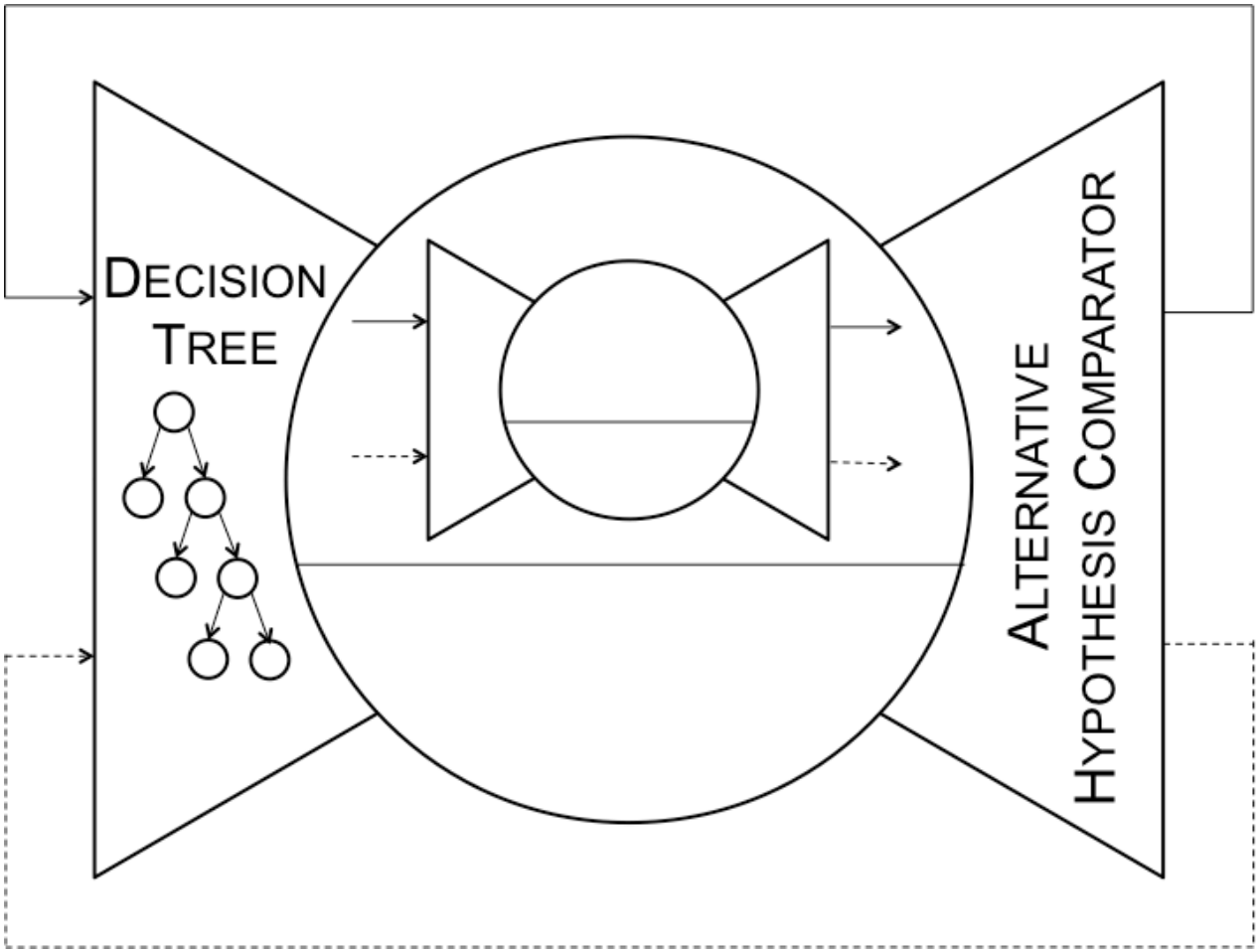
Parallel/Iterative



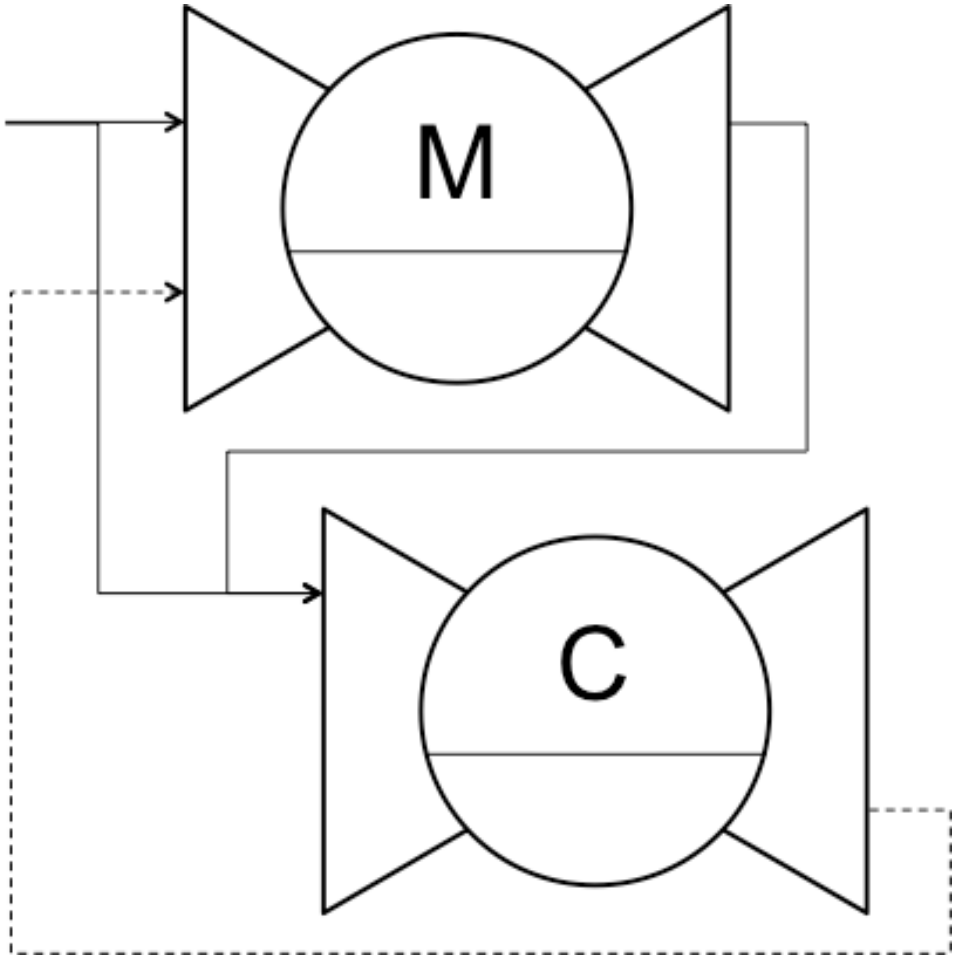
Map-Reduce



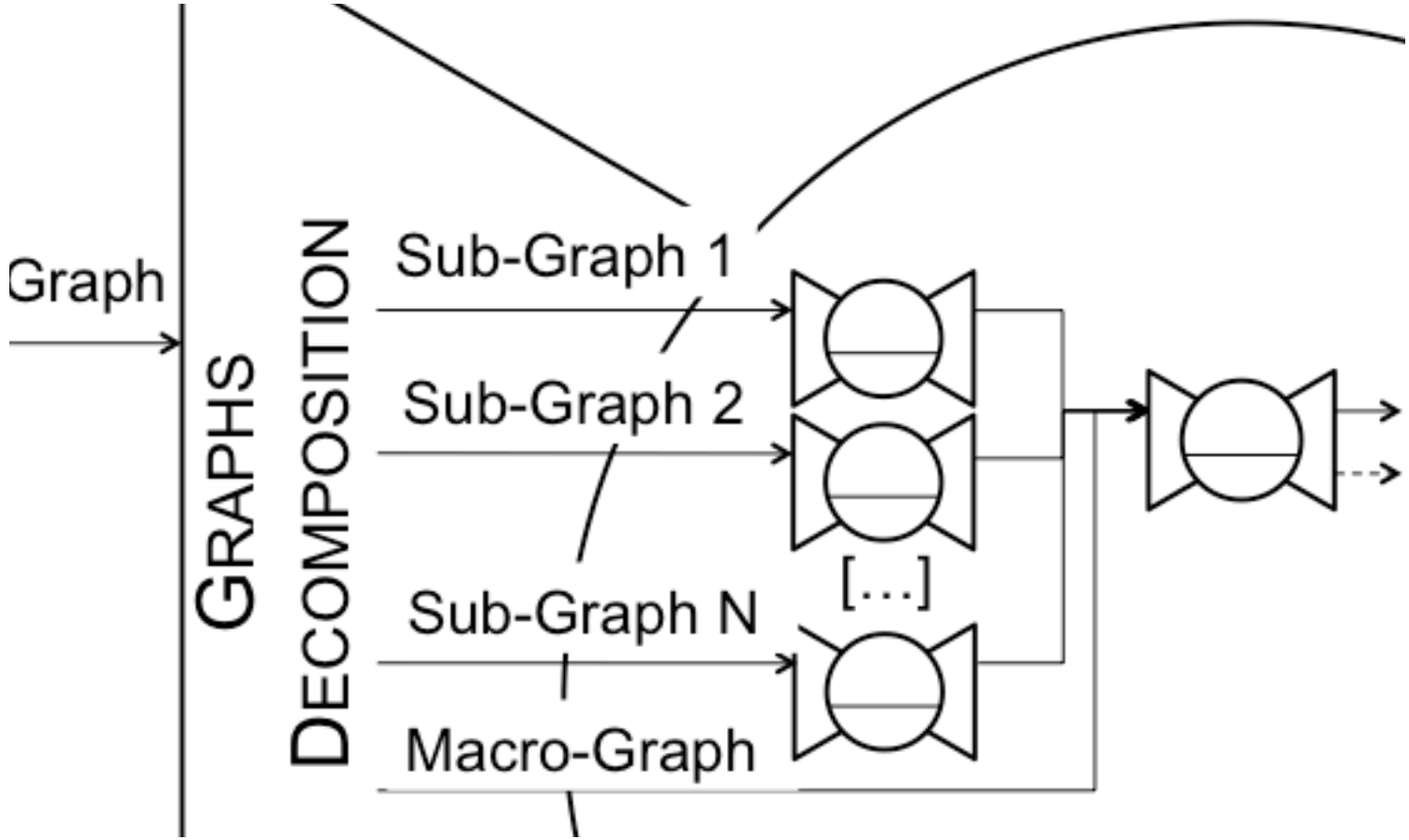
What-If



Drift Control



Graph Decomposition

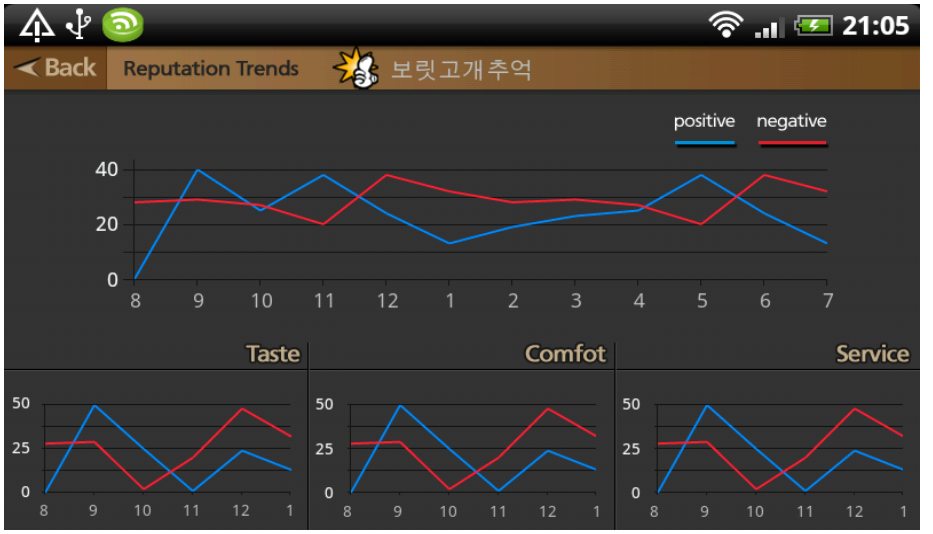


Summary of ICT Requirements for Scientific Big Data Management

- In the "small" (modules, each processing terabytes of data)
  - Identify reusable data formats as pattern types
  - Identify reusable computations as data analysis models
  - Identify appropriate data transformations for data preparation
  - Identify appropriate quality assessments for data evaluation
- In the "large" (composing mega-modules)
  - Foster composition through appropriate composition abstractions + infrastructures
  - Allow for assessing properties of the mega-module composition
    - Correctness, reliability, etc.
  - Allow for inspection of mega-modules during processing
    - Assessing current state, intermediate results, etc.
  - Allow for dynamic reconfiguration of each mega-module
    - Scale up and down in response to the load, recover a computation after a fault, etc.

Examples of applications through compositions of MegaModules

BOTTARI: restaurant recommender based on geo-aware social media analytics



BOTTARI as a Mega-Model Composition

- Explicit module structure with input-output relationships

BOTTARI

Outputs

Inputs

Geo-Spatial Model

Predictive Model

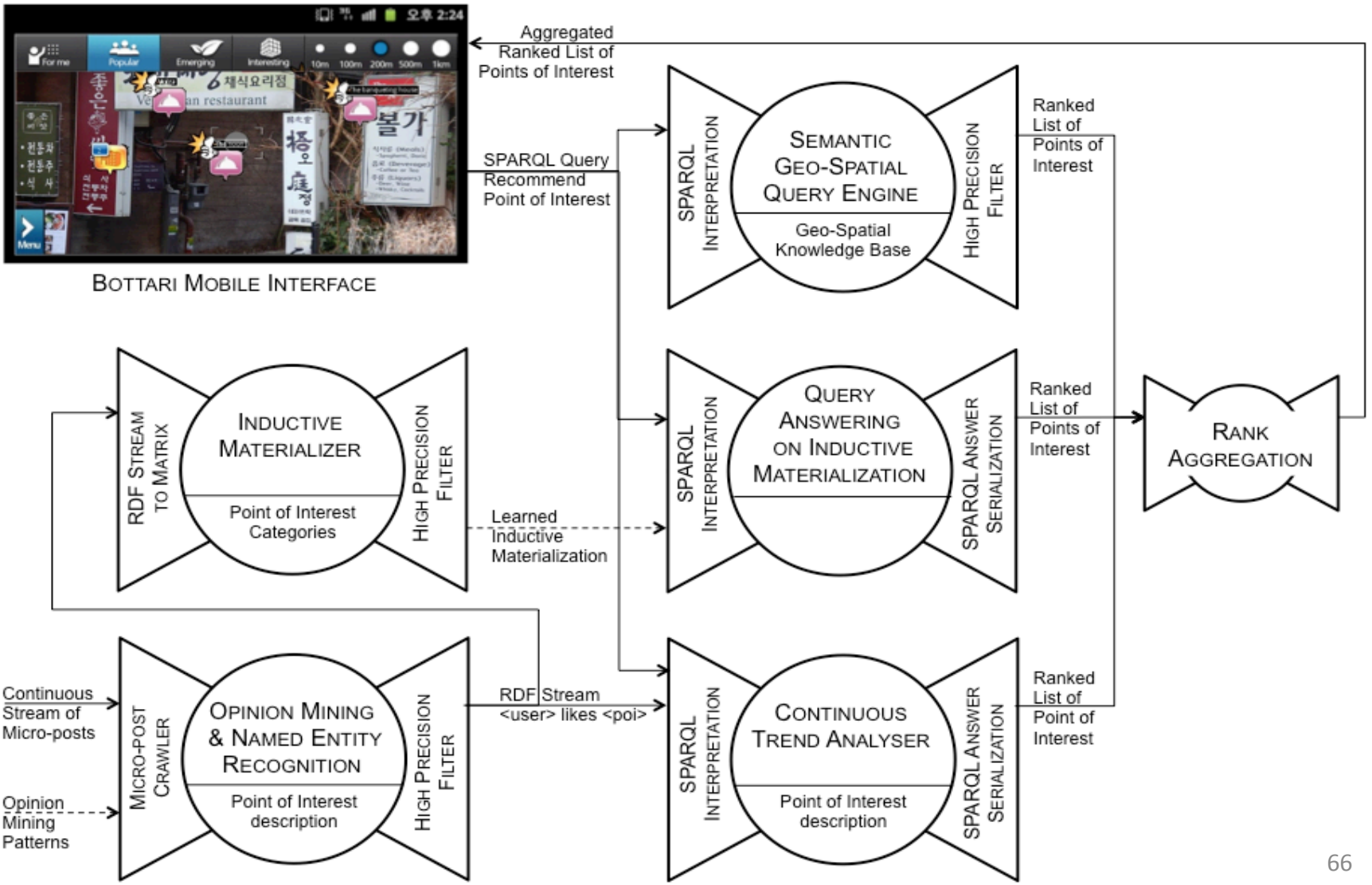
Temporal Model

Social Media Crawler and Miner

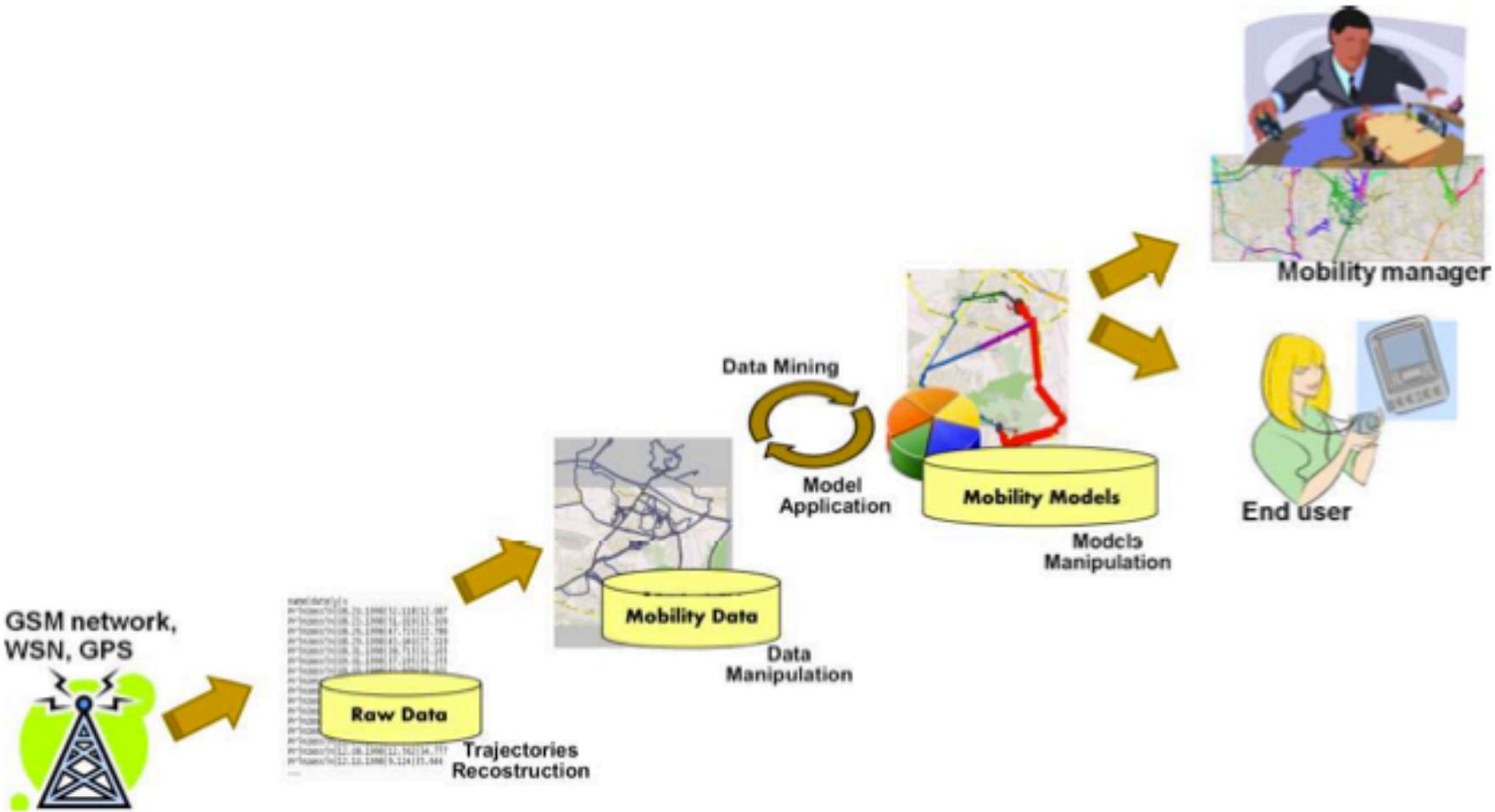
BOTTARI Models

- *Geo-spatial model*
  - Input: User position, semantic + geo-spatial description of restaurants
  - Output: a list of matching restaurants ranked by distance from the user
- *Temporal model*
  - Input: stream of liked restaurants
  - Output: ranking of restaurants in "like" order in the last week/month/ quarter
- *Predictive model*
  - Input: materialized stream of liked restaurants
  - Output: prediction of the restaurant which will be chosen by the user as best-fit
- *Social Media Crawler and Miner*
  - Input: stream of tweets of people about restaurants
  - Output: stream of most liked restaurant after named entity recognition and sentiment mining

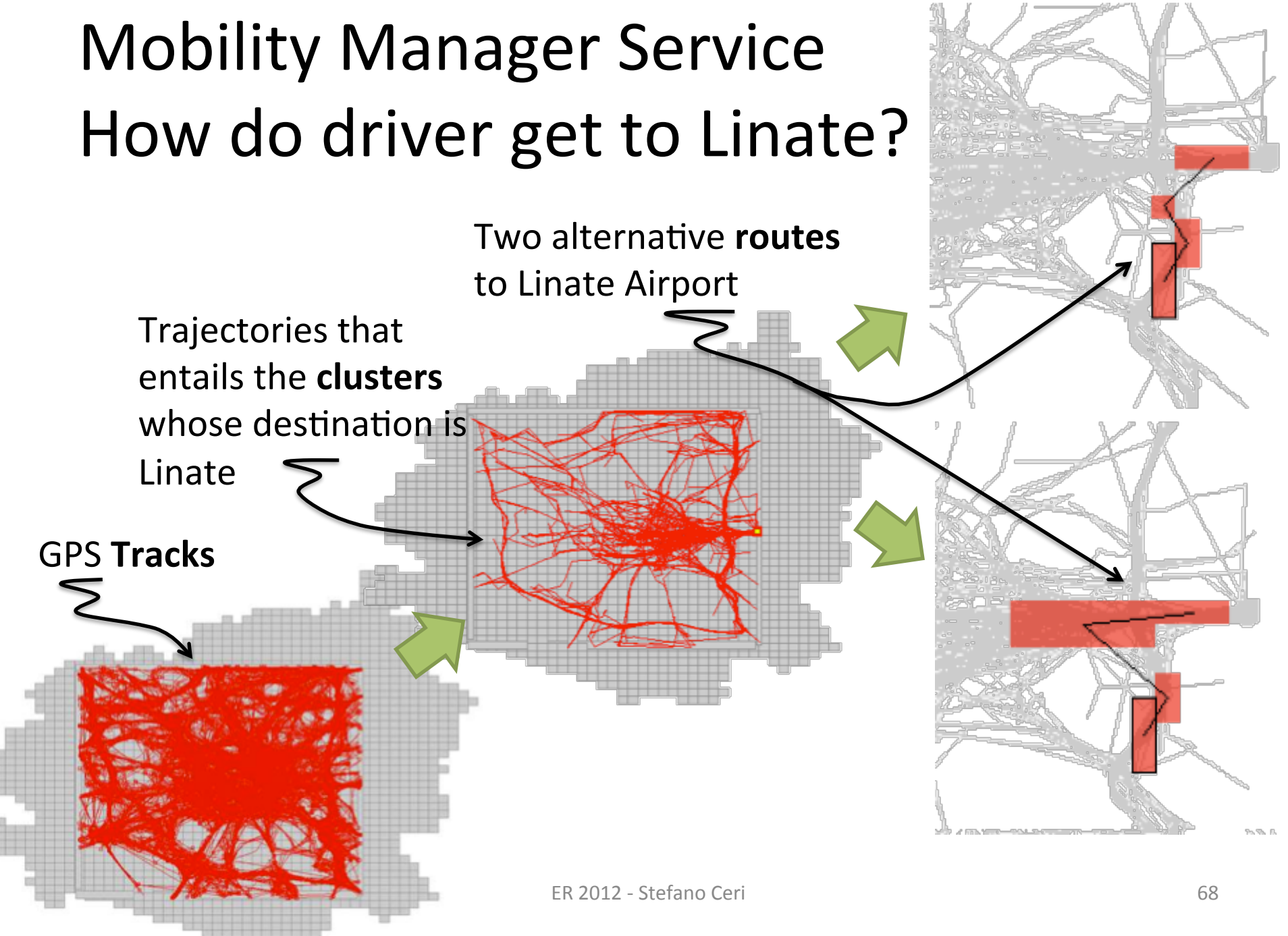
Mega-modularization of Bottari



Mobility analysis system

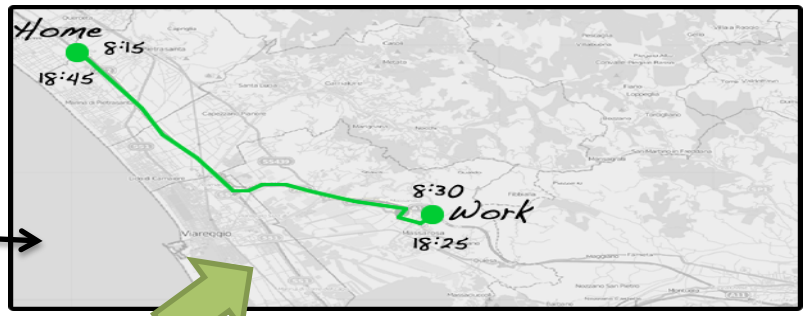


Mobility Manager Service
How do driver get to Linate?

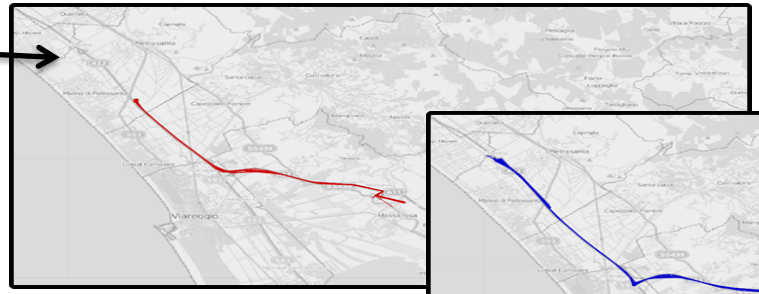


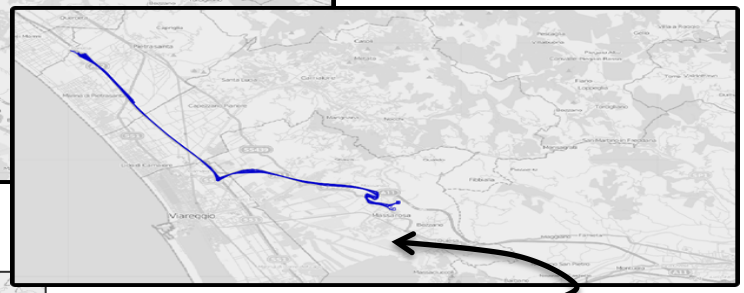
End-User Service
User's Mobility Profiling for Car Pooling

Spatio-Temporal
User's **mobility profile**

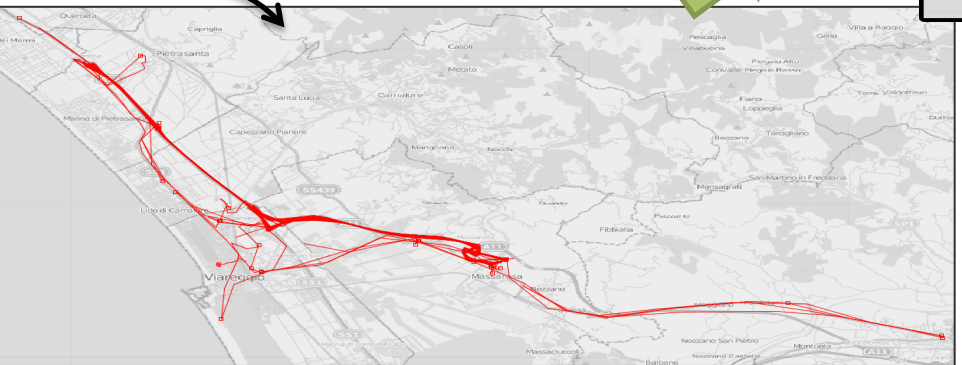


Trajectories that entail the **cluster** "Home-Work"





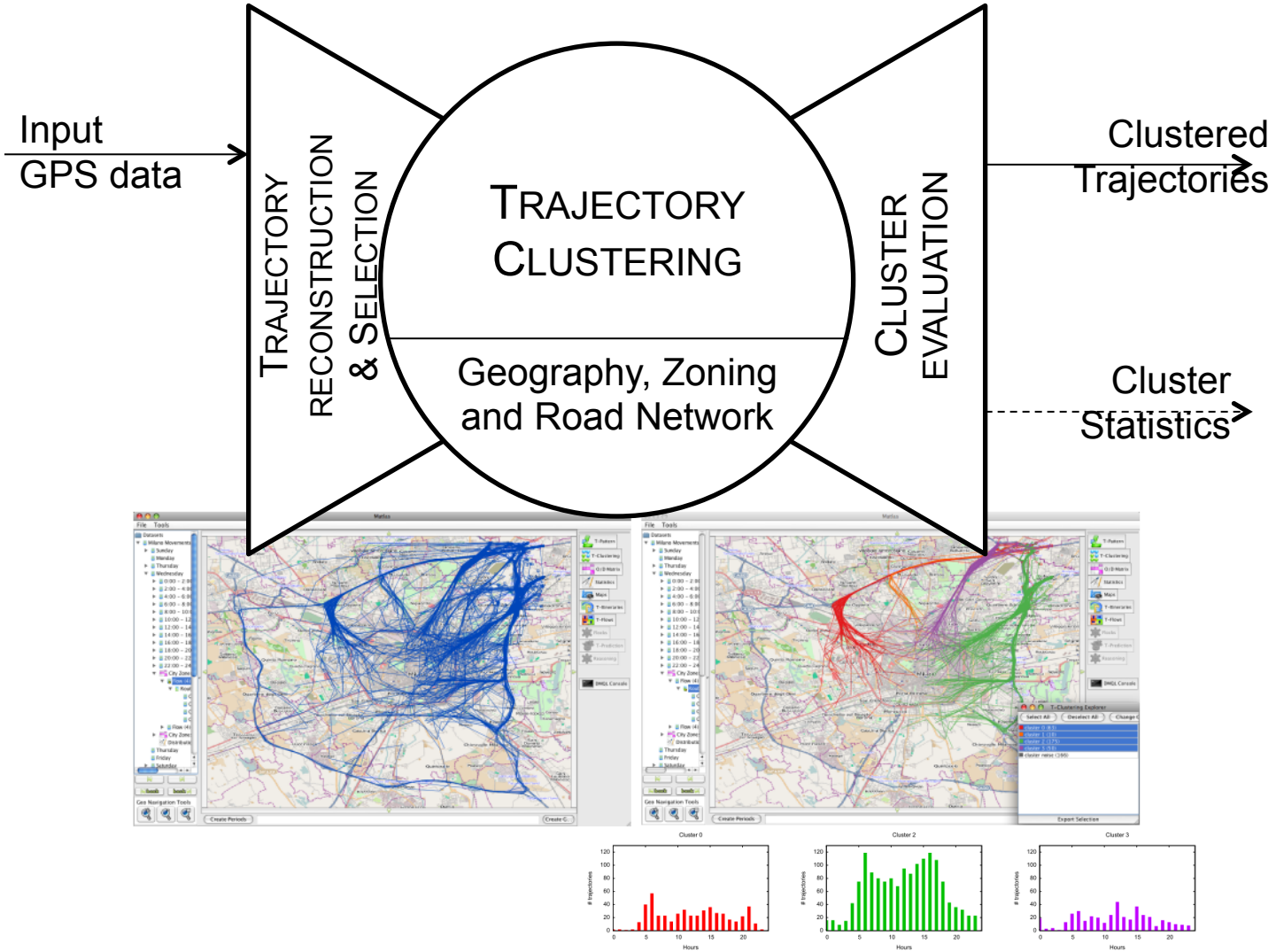
User's GPS **Tracks**



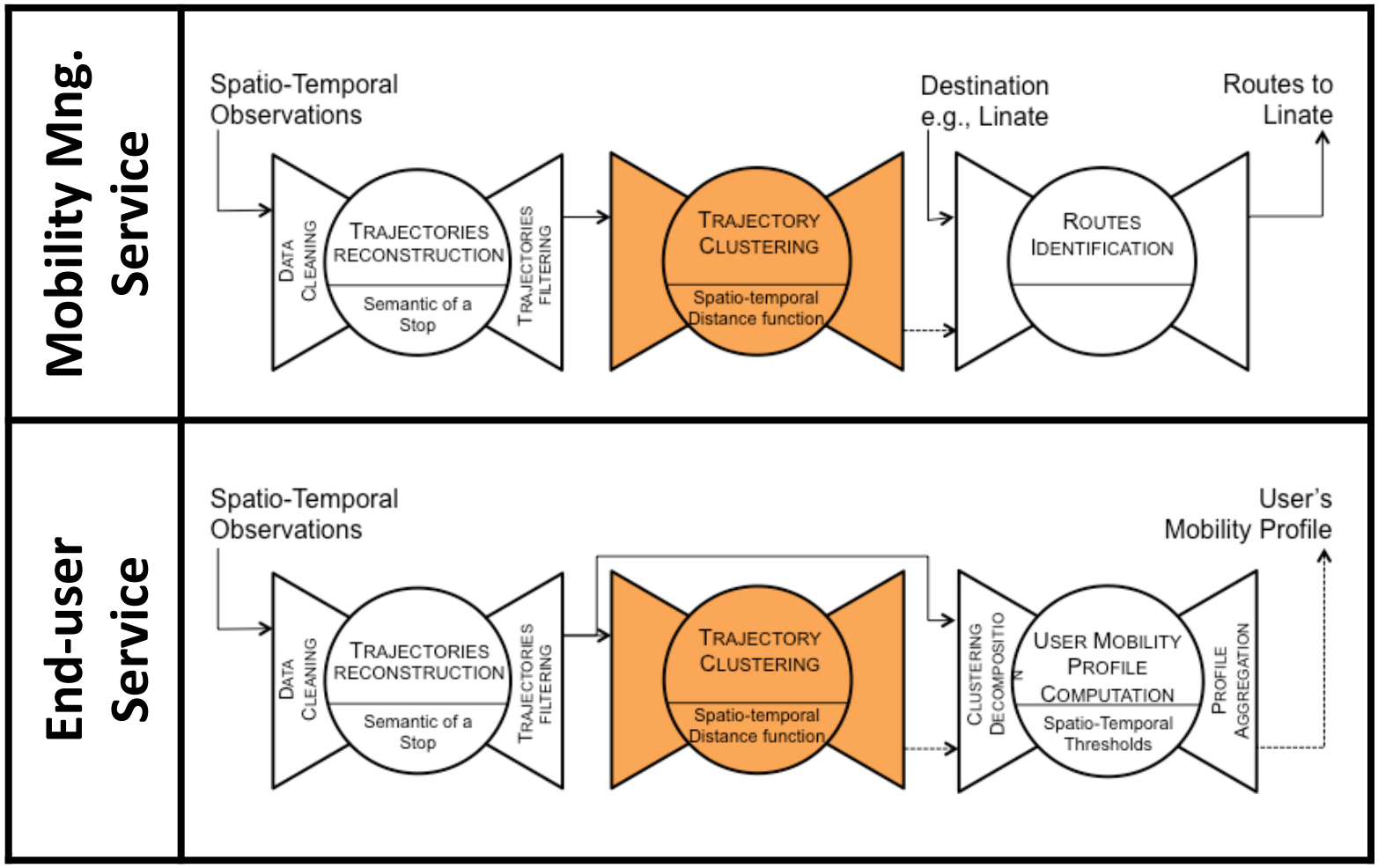
Trajectories that entail the **cluster** "Work-Home"

Home = most frequent location
Work = second most frequent location

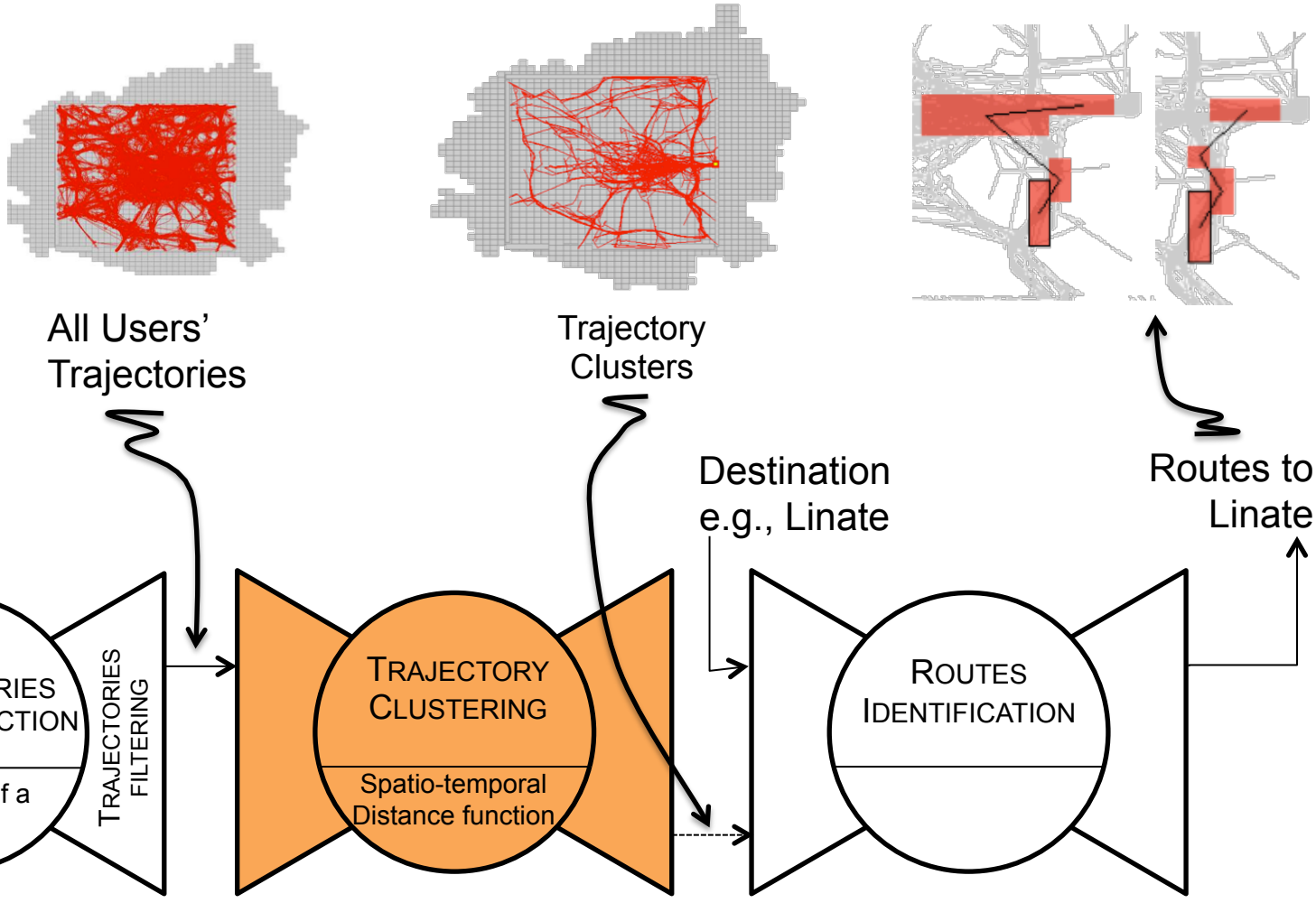
Mega-modularization of Trajectory Clustering



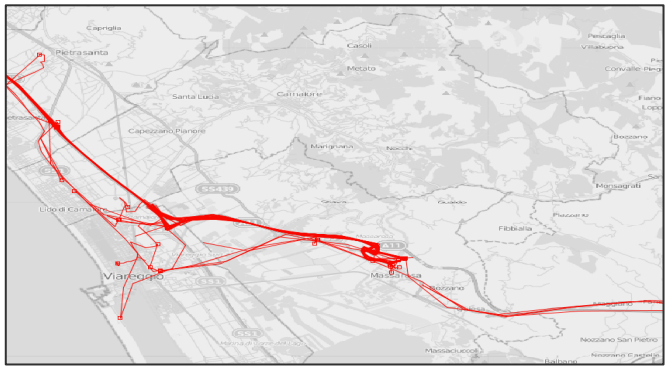
Trajectory Clustering Megamodule Usages



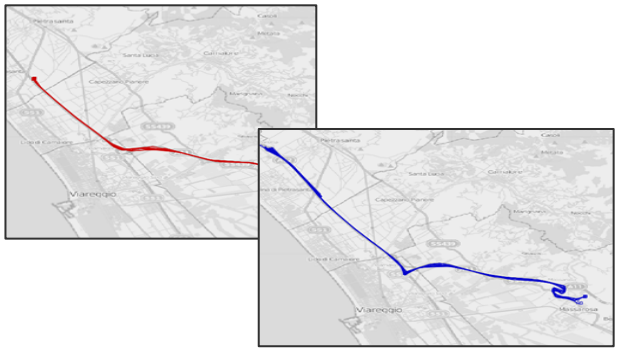
Mega-modularization for Mobility Manager Service



Mega-modularization of Trajectory Clustering for Car Pooling



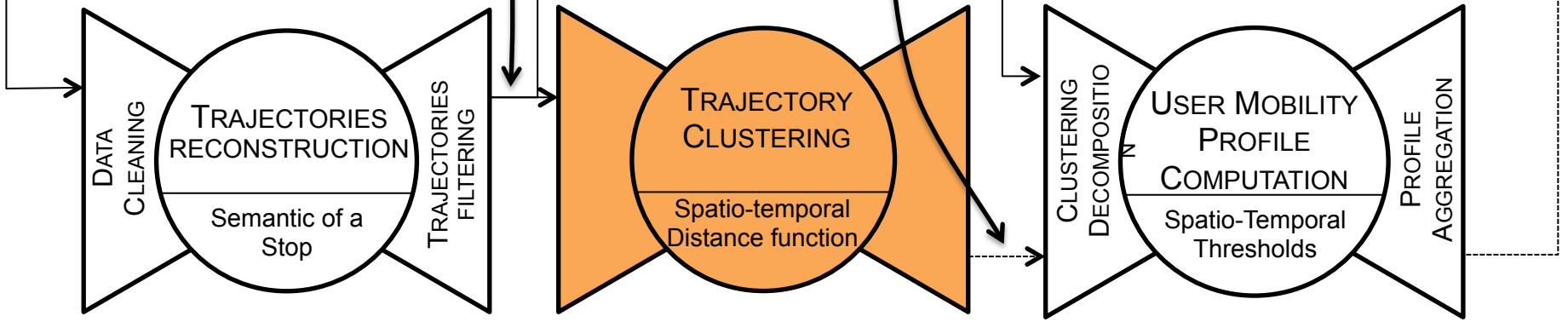
Single User's Trajectories



Single User's Trajectory Clusters

User's Mobility Profile

Spatio-Temporal Observations



Research questions & agenda

- Express a large collection of patterns through suitable (relational) language extensions
- Build an ontology of mega-models, support reasoning upon the ontology for deriving properties of mega-models
- Define/classify composition abstractions and define the mega-modeling composition language
- Consider research problems related to:
  - Optimization (inter vs intra)
  - Orchestration
  - Inspection
  - Adaptation
- Build the software engineering tools and environment for building and composing mega-models

Summary of the talk

- Motivations
  - Examples of big scientific data, FuturICT
  - Typical research questions
- Why MegaModelling?
  - History of the term
  - What should be solved
- What is a pattern
  - Application-independent , tabular, composable
- What is a mega-module
  - Ingredients: Preparation / Analysis / Evaluation
  - Composition abstractions
- Examples of mega-modularizations
- To-do list