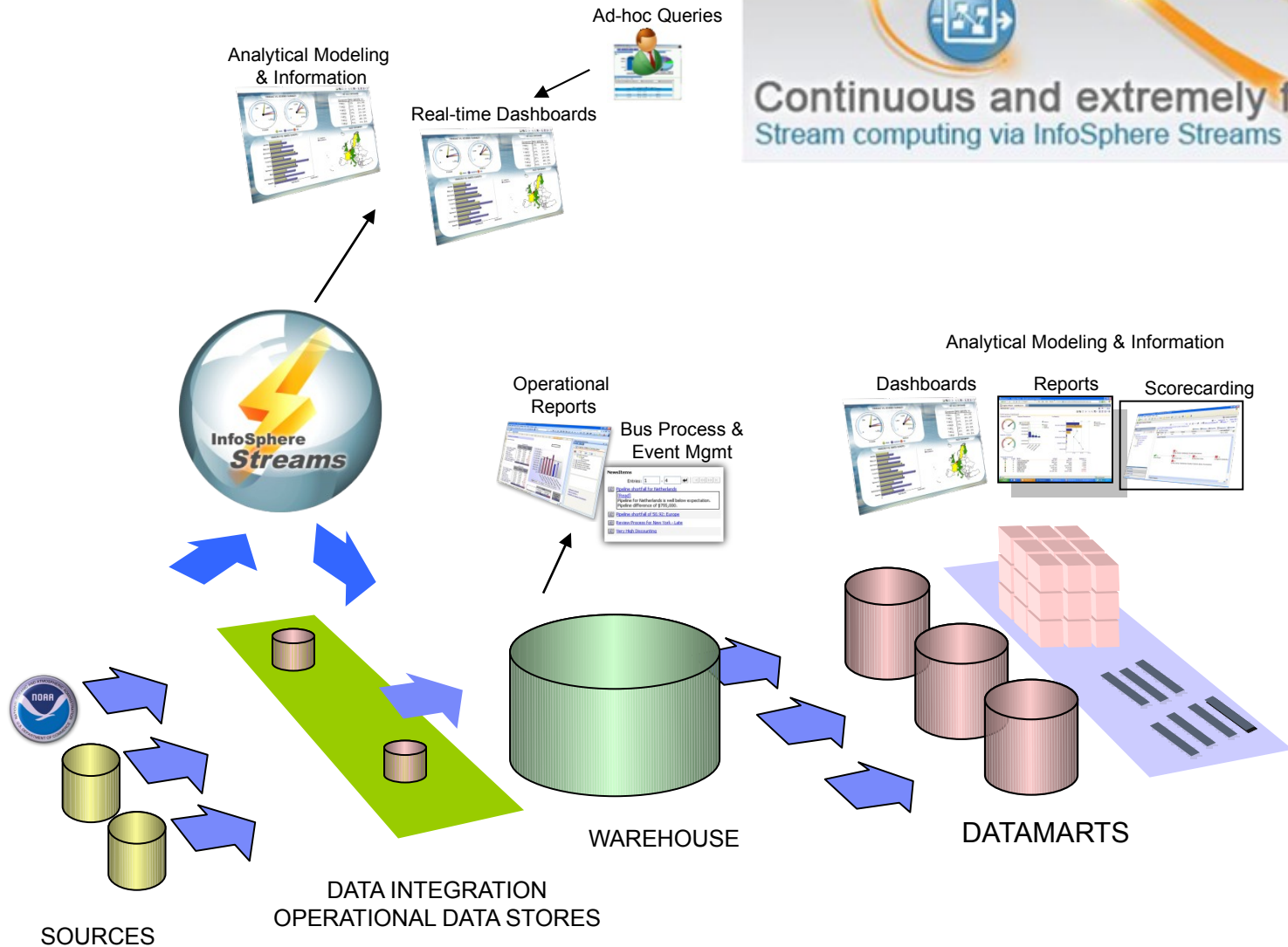


# Knowledge Engineering for Planning-based Data-flow Composition

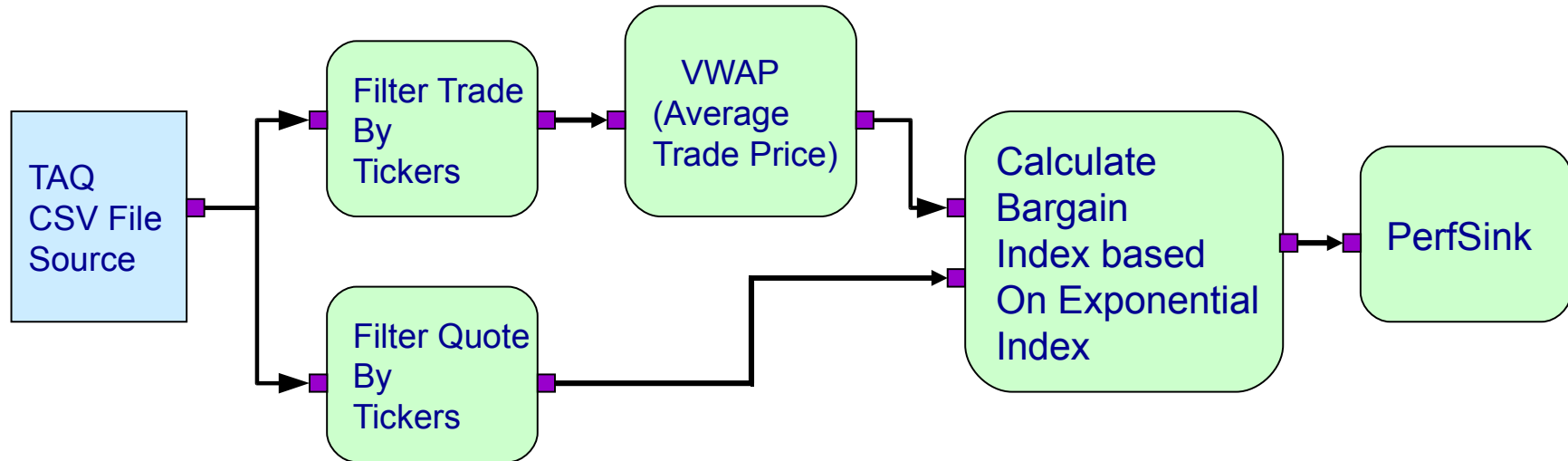
Mark D. Feblowitz, Anand Ranganathan, Anton V. Riabov, and Octavian Udrea  
IBM T. J. Watson Research Center  
Yorktown Heights, NY, USA

ICKEPS, 6/25/2012

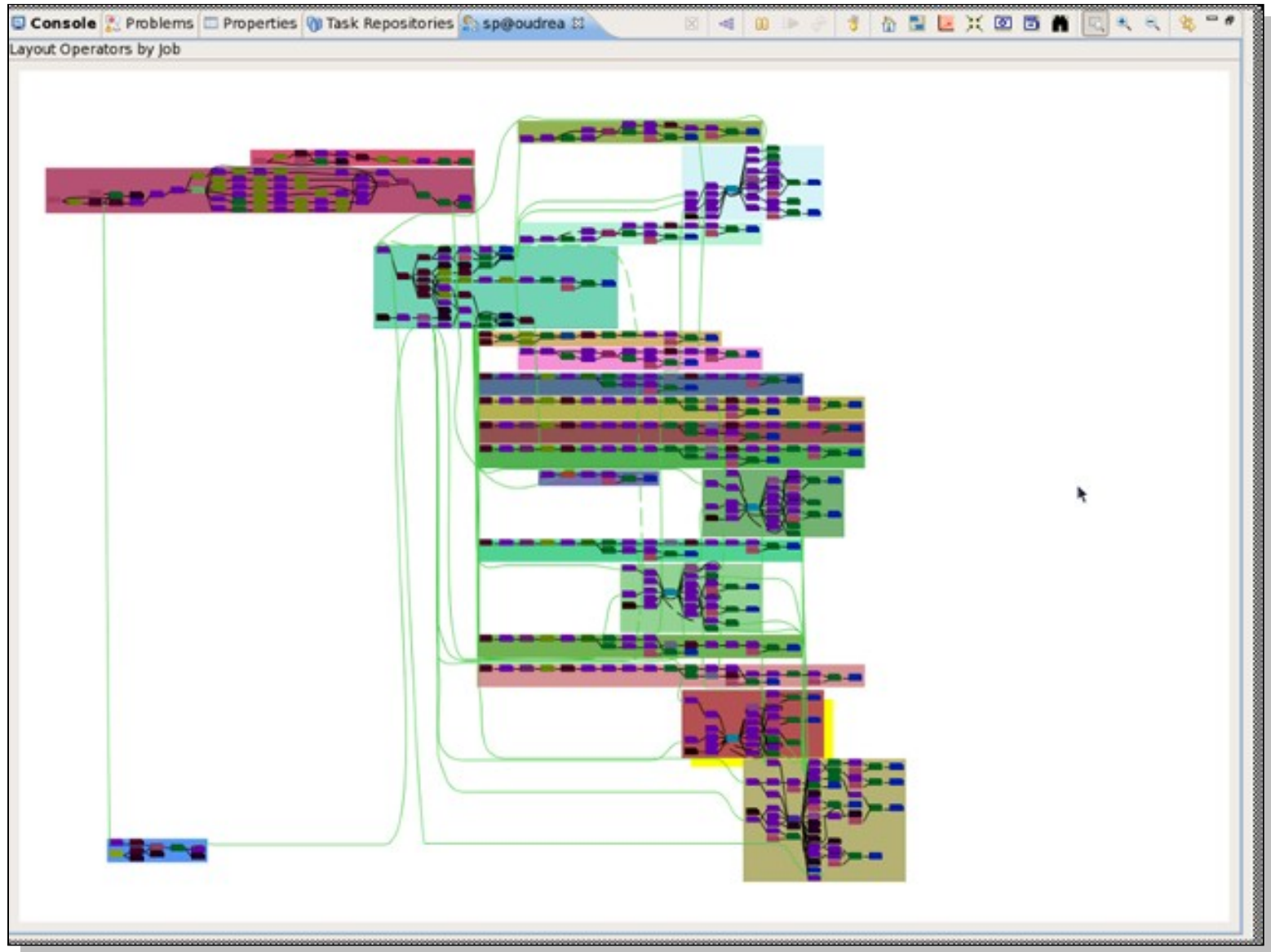
# Background: Distributed Stream Processing



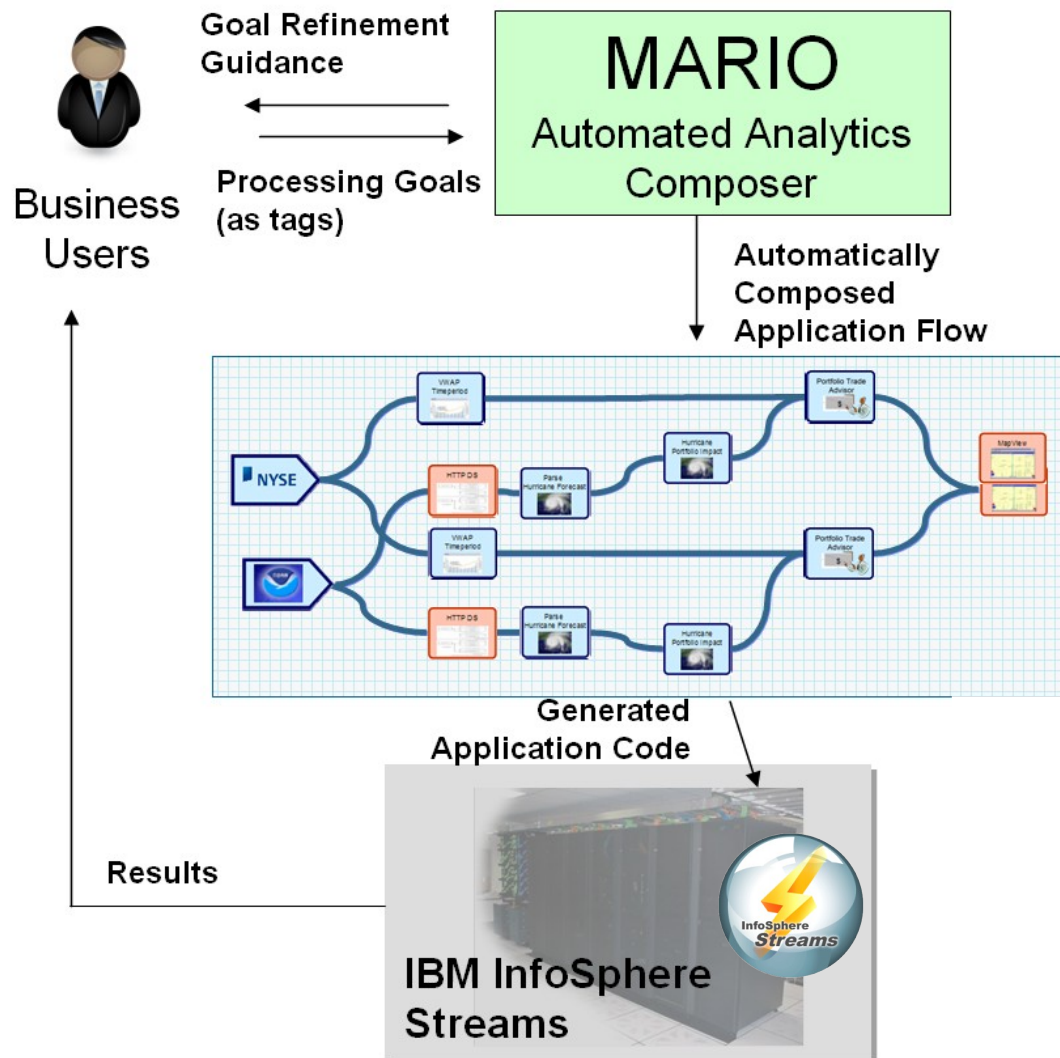
# A Sample Stream Processing Application



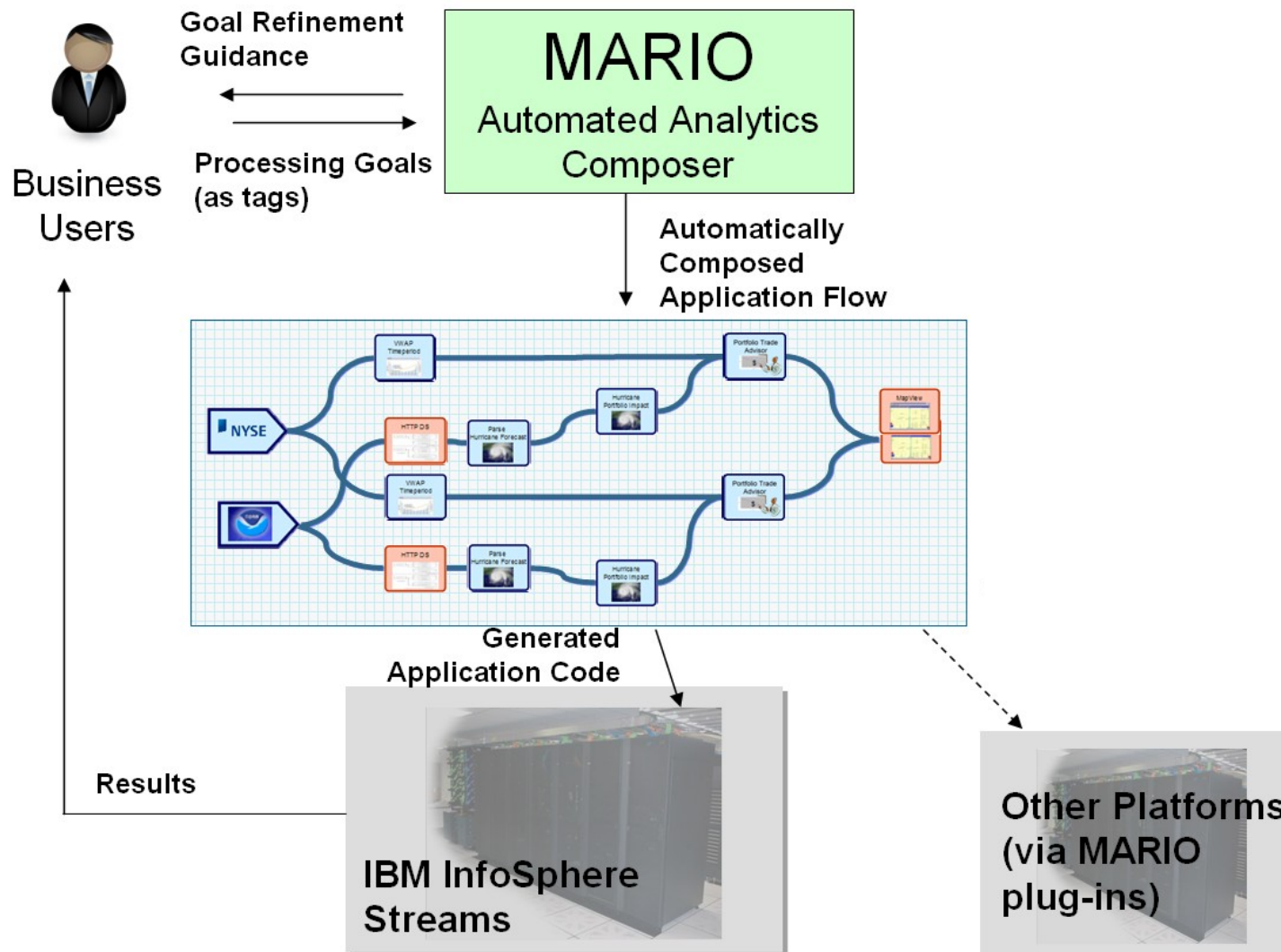
# Stream Processing: A More Complex Example



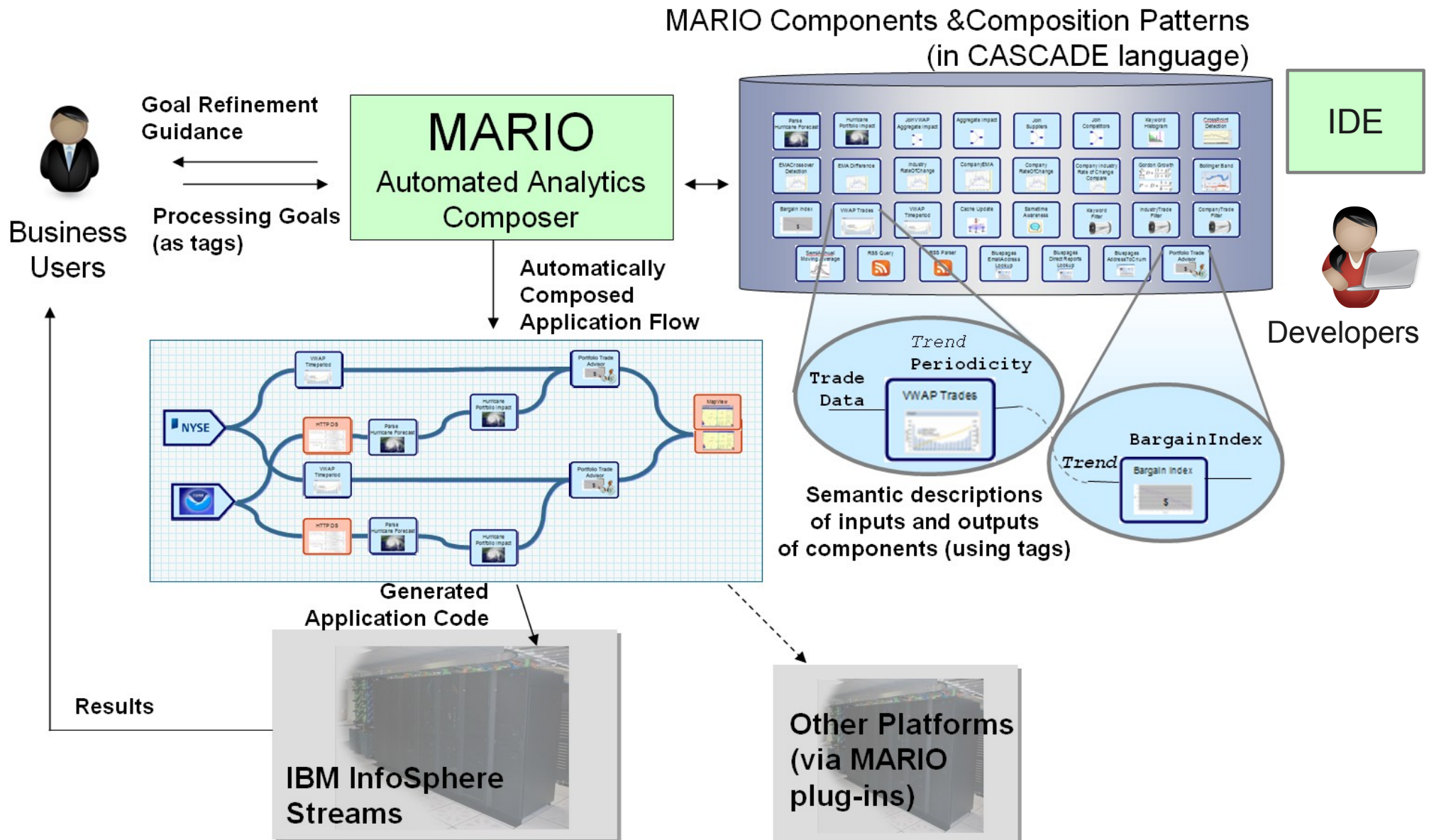
# MARIO Planner and Cascade Language



# MARIO Planner and Cascade Language



# MARIO Planner and Cascade Language



# Pilot Deployment

- Deployed by a customer, in a pilot deployment since 2009
  - Pilot includes Cascade IDE, MARIO, InfoSphere Streams
  - Developers use Eclipse IDE to describe components, composition patterns, etc
  - End-users compose, launch jobs and view results using MARIO Web-based interface



# Related Work

- Mashups and Visual Programming (e.g., drag-and-drop) tools
  - For end-users, the interaction with MARIO is goal-centric – there is no flow editor in MARIO
  - Developers describe components and composition patterns in text, using Cascade language
- Web Service Composition (e.g., using planning)
  - Our approach can be used to compose web services in some scenarios, but there are differences from other approaches

# Specifying Goals and Planning Domains

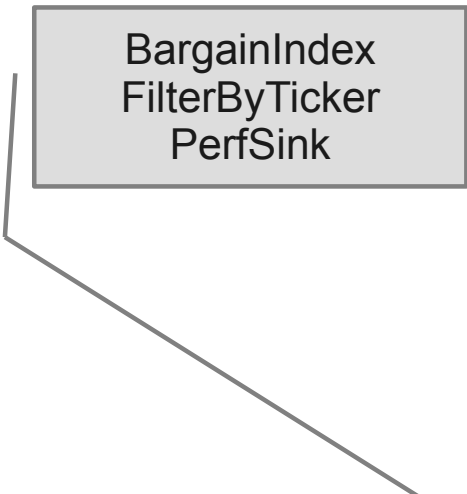
- In historical order, we have experimented with:
  - PDDL
  - SPPL [[AAAI'05](#), [ICAPS'06](#)]
    - Exponentially faster performance compared to PDDL
  - OWL-based semantic annotations [[ICWS'07](#)]
    - Rich semantic descriptions
  - Tags and tag taxonomies [[WWW'08](#), [OOPSLA'08](#)]
    - No OWL skills required to describe components and specify goals
  - Cascade [[CIKM'09](#)]
    - Modular structure, formally defined flow patterns

# Specifying Goals and Planning Domains

- Current implementation **includes**:
  - PDDL
  - ➔ **SPPL** [AAAI'05, ICAPS'06]
    - Exponentially faster performance compared to PDDL
  - OWL-based semantic annotations [ICWS'07]
    - Rich semantic descriptions
  - ➔ **Tags and tag taxonomies** [WWW'08, OOPSLA'08]
    - No OWL skills required to describe components and specify goals
  - ➔ **Cascade** [CIKM'09]
    - Modular structure, formally defined flow patterns

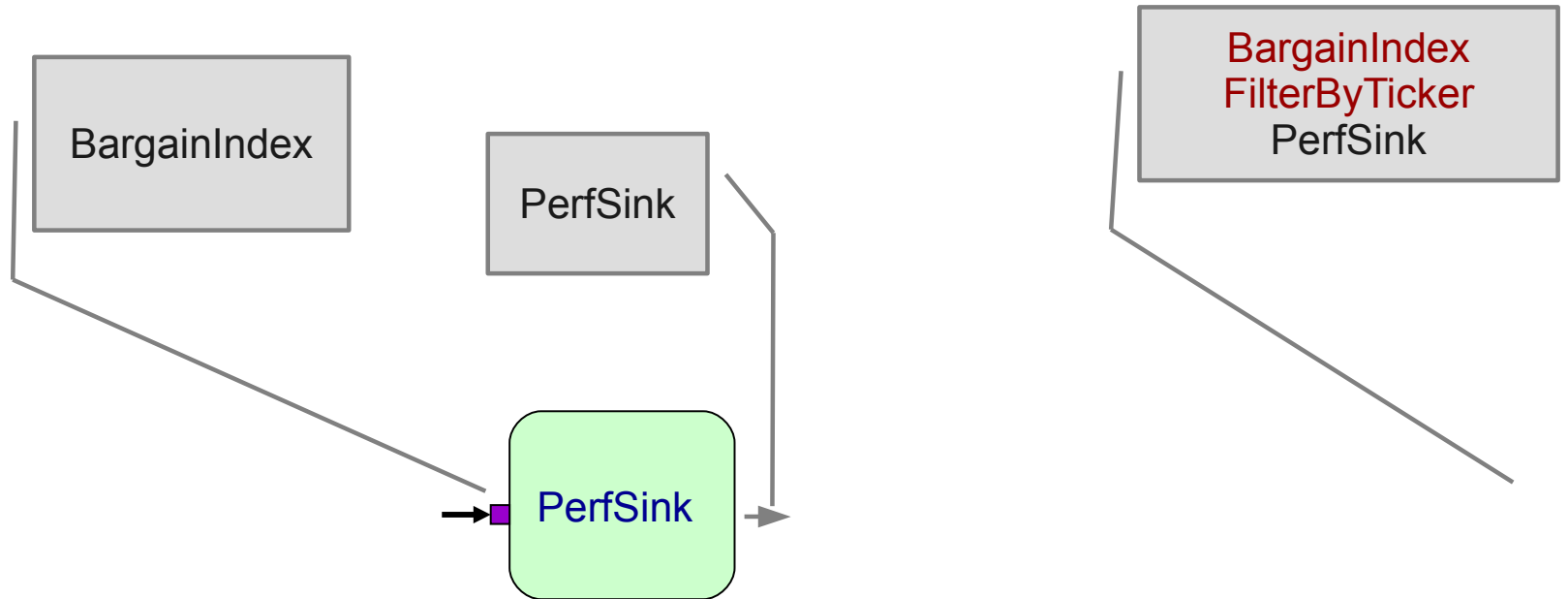
# Example

Goal:



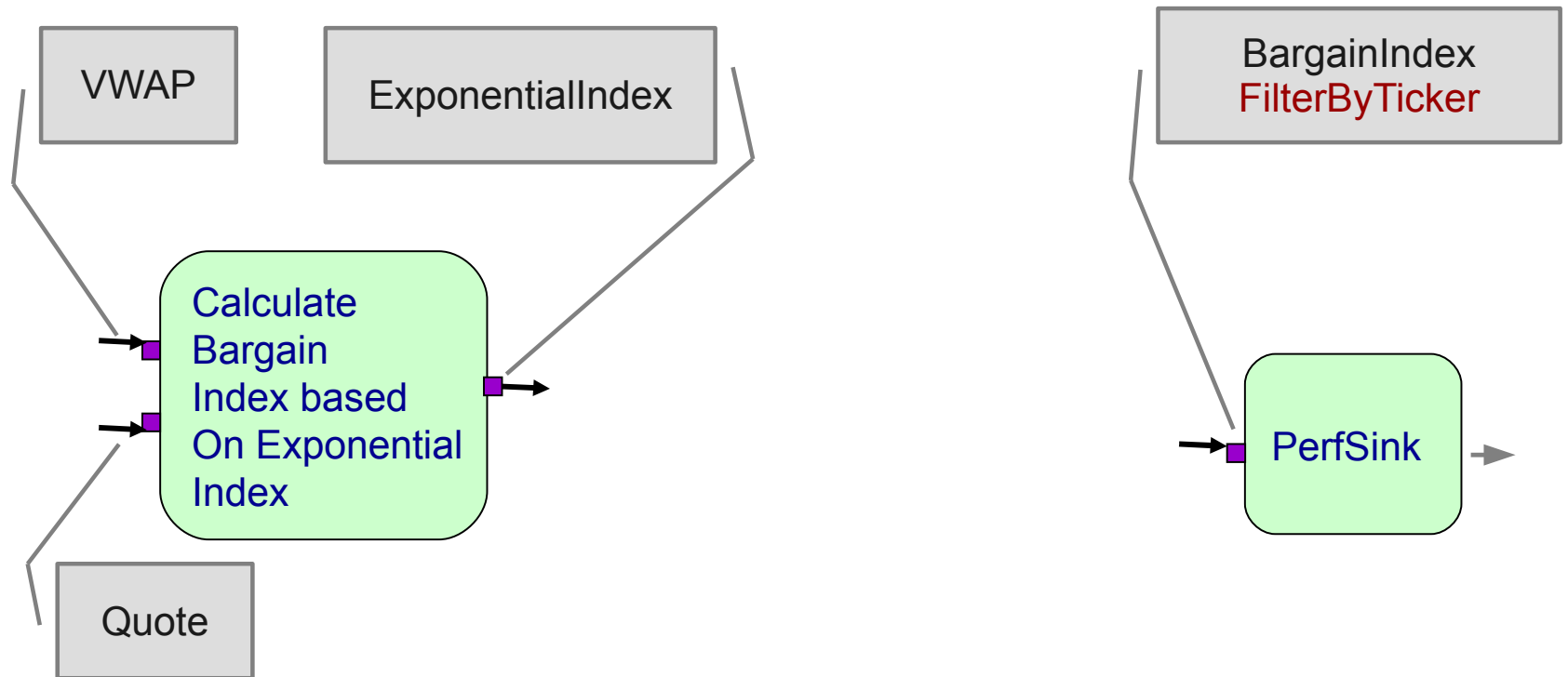
BargainIndex  
FilterByTicker  
PerfSink

# Example

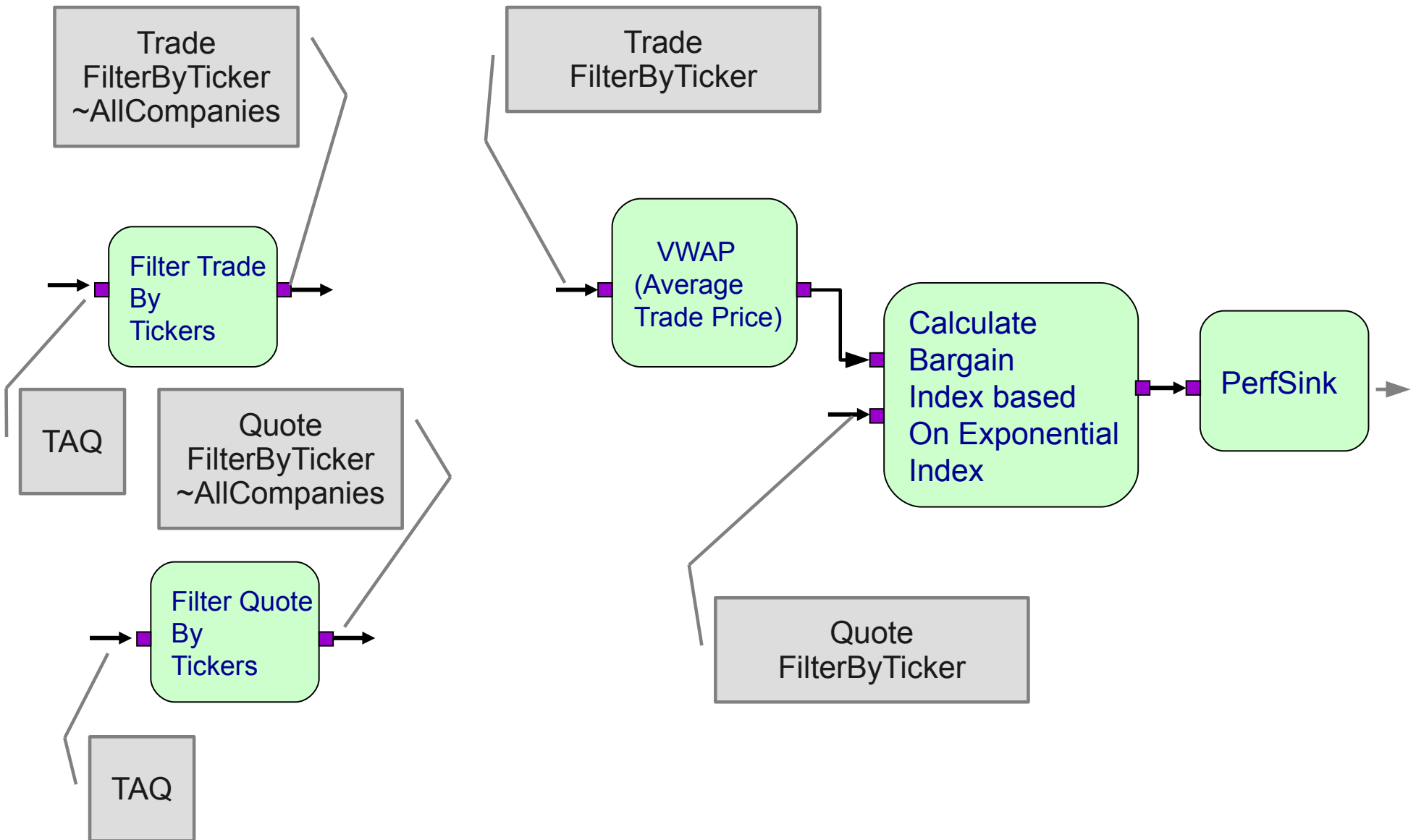


# Example

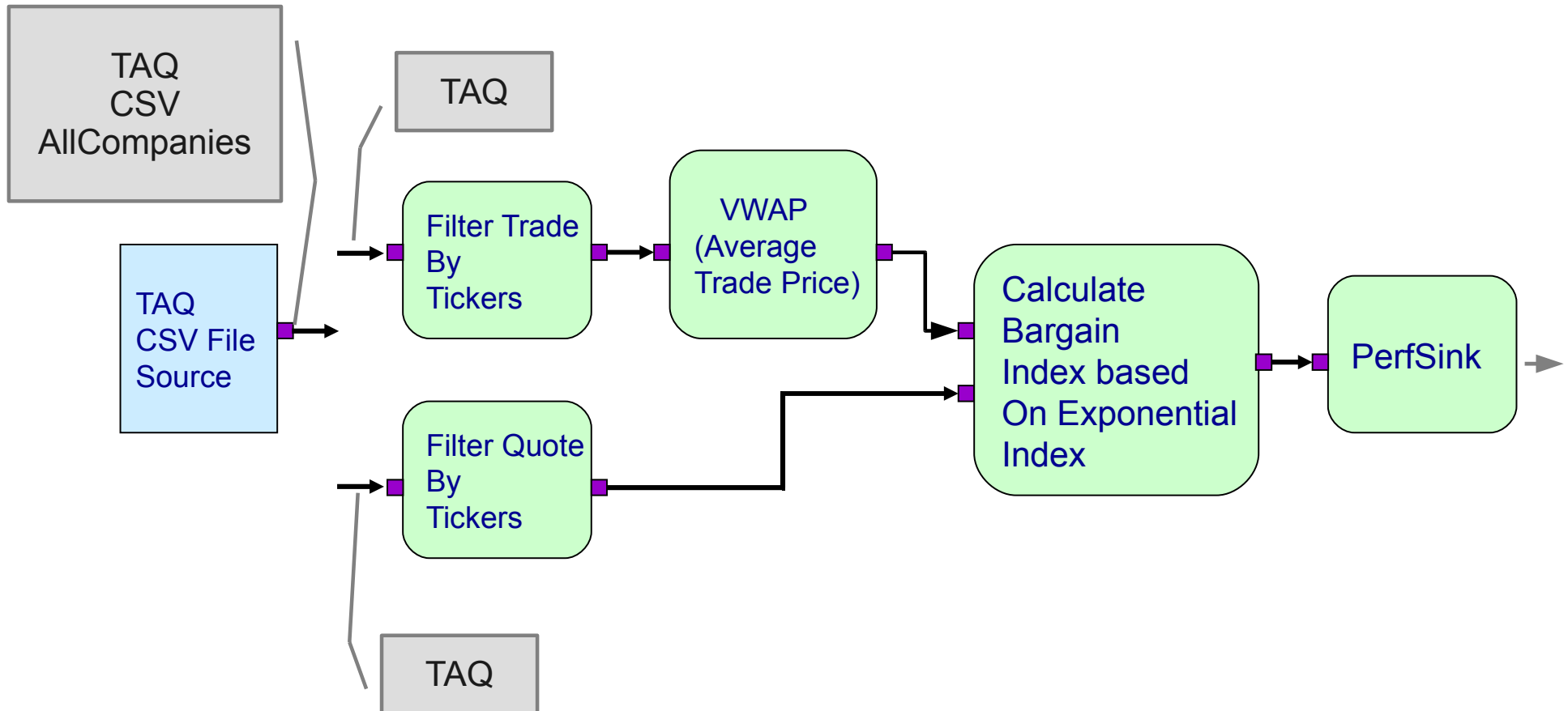
Taxonomy: ExponentialIndex  $\rightarrow$  BargainIndex



# Example

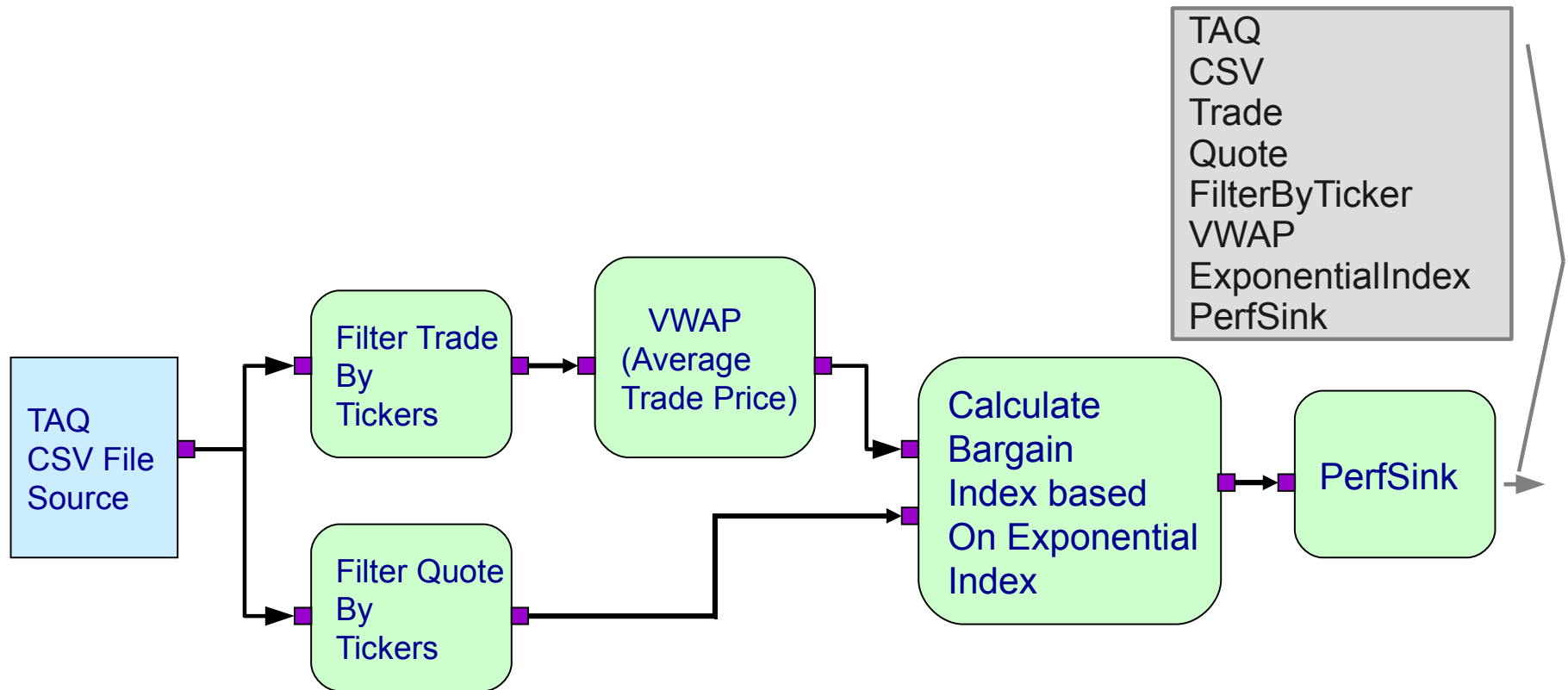


# Example

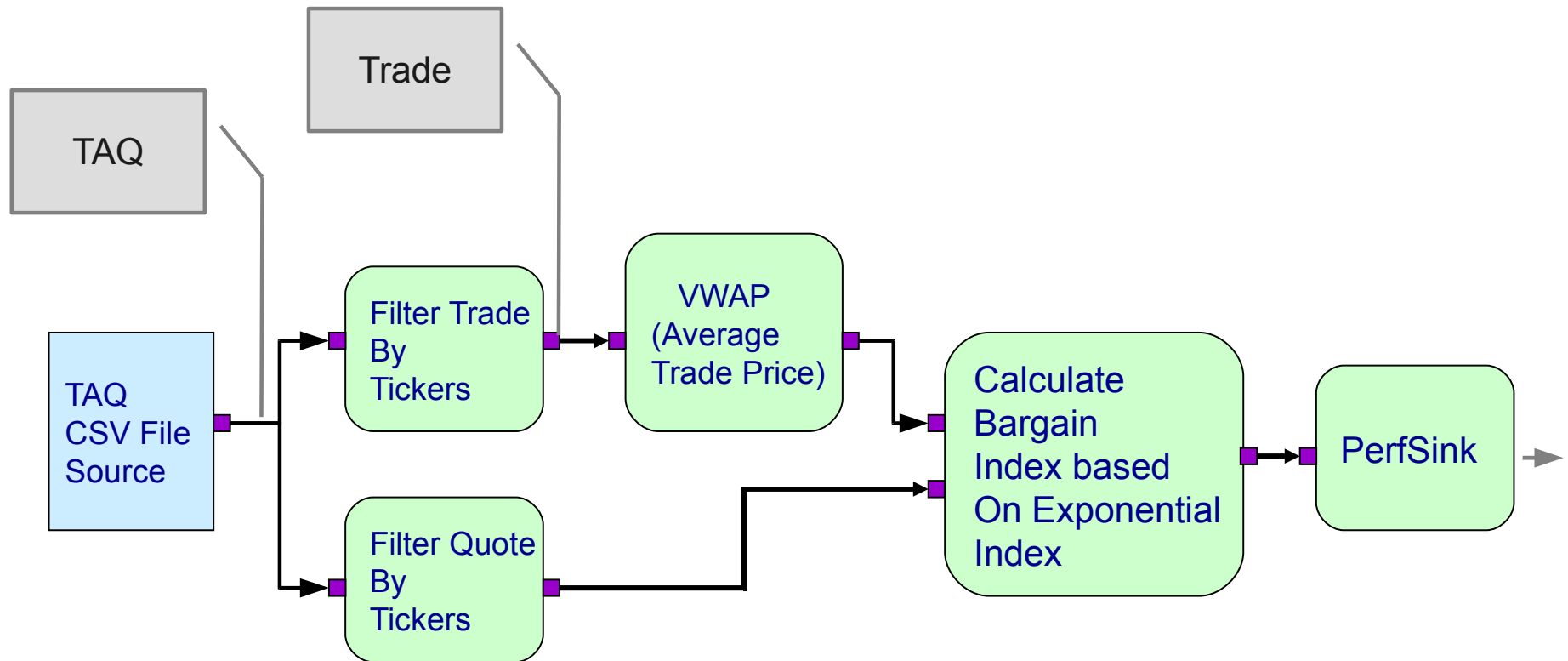




# Example

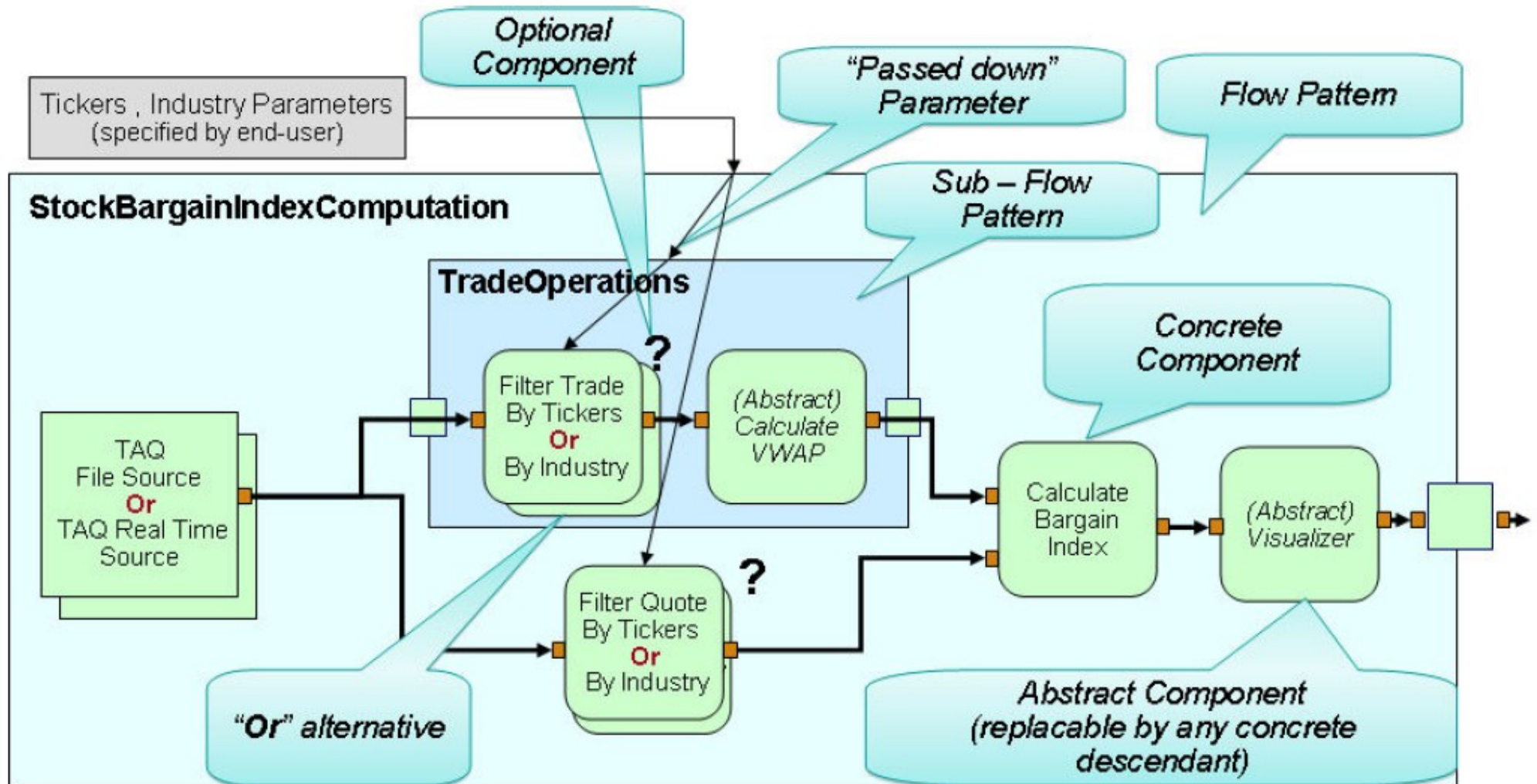


# Adding Another Filter?



How many new plans can be generated if we add one more component?

# Cascade Flow Patterns



# From Cascade to SPPL (and Back)

- Cascade compiler:
  - Parses and validates Cascade
  - Translates Cascade to input/output tag descriptions
- MARIO Tag-based Planner:
  - Translates tag component descriptions, goals and tag taxonomies to SPPL
  - Runs SPPL planner
  - Translates resulting plans into flows, annotates them with tags

# Cascade Component Example

```
/*  
@type "spl"  
@title "Filter Trade and Quote By Tickers"  
@tags TradeQuoteFilter ByMonitoredTickers ~AllCompanies OneCompany  
*/  
component FilterTradeQuoteByTickers(output TradeQuoteFilter; input TradeQuote) {  
    param string $monitoredTicker;  
    /$  
    stream <rstring ticker, rstring industry, rstring date, rstring time, rstring ttype,  
        decimal64 price, decimal64 volume, decimal64 vwap, decimal64 bidprice,  
        decimal64 askprice, decimal64 asksizes> @TradeQuoteFilter@  
    = Functor(@TradeQuote@) { param filter: ticker == "@monitoredTicker@"; }  
    $/  
}  
  
composite StockBargainIndexComputation(output final) {  
    param string $monitoredTickers ; string $industry;  
    graph  
    // ...  
    stream FilteredTradeQuote? =  
        FilterTradeQuoteByTickers(TradeQuote) {param monitoredTicker: $monitoredTickers;}  
        | FilterTradeQuoteByIndustry(TradeQuote) {param industry: $industry;}  
    // ...  
}
```

# Generated SPPL

```
(:action FilterTradeQuoteByTickers--StockBargainIndexComputation-VWAP-FilteredTradeQuote-3950_c
:singleton
:parameters ( ?id1_p - __TradeQuote-StockBargainIndexComputation-VWAP_t
              ?id2_p - __VWAP-FilteredTradeQuote-3950_t
              ?id3_p - __StockBargainIndexComputation_t ?id4_p - _monitoredTicker_t)
:cost ( -1 1)
:precondition [TradeQuote] (and (T ?id1_p) )
:precondition [monitoredTicker] (and (T ?id2_p) (T ?id3_p) (T ?id4_p) )
:effect [TradeQuoteFilter] (and
  (S ByMonitoredTickers_t) (S Complex_t) (S OneCompany_t)
  (T __FilteredTradeQuote-StockBargainIndexComputation-VWAP_t)
  (not (S AllCompanies_t)) )
)
(:action Dummy-FilterTradeQuoteByTickers-StockBargainIndexComputation-VWAP-3951_c
:singleton
:parameters ( ?id1_p - __TradeQuote-StockBargainIndexComputation-VWAP_t)
:cost ( -0.00001 1)
:precondition [TradeQuote] (and (T ?id1_p) )
:effect [TradeQuoteFilter] (and
  (T __FilteredTradeQuote-StockBargainIndexComputation-VWAP_t)
  (T __TradeQuote-StockBargainIndexComputation-VWAP_t)
)
)
)
```

# Eclipse-Based Cascade IDE

The screenshot displays the Eclipse IDE interface with the Cascade DSL code editor open. The code defines a graph for stock computation. Callouts highlight specific features:

- Input parameter with a default value:** Points to the `industry: UserParam("Industry", "|Tech|Finance|Transport|");` line in the `StockBargainIndexComputation()` function.
- Optional component (in this case, implemented by a choice):** Points to the `stream FilteredTradeQuote? = FilterTradeQuoteByTickers(TradeQuote) {param monitoredTicker: $monitoredT} | FilterTradeQuoteByIndustry(TradeQuote) {param industry: $industry;}` line.
- Choice construct:** Points to the `| TableView(BargainIndex) { } | StreamPlot(BargainIndex) { }` part of the final stream definition.
- Subgraph composite:** Points to the `composite BIComputationCore(input FilteredTQ; output BI) {` block.

# MARIO User Interface

The screenshot displays the MARIO User Interface within a Mozilla Firefox browser window. The main content area features a flow graph with a central vertical loop and horizontal branches, composed of various nodes and connectors. To the left, a sidebar contains a 'Current Tag Selection' section with 'TableView' selected, and a 'Sources' section with 'FileSource', 'TAQ', and 'TcpSource' listed. Below this is an 'Analysis' section with 'Filters' and 'Other' options. On the right, a 'Compute Complex Bargain Index' section is visible, including a 'Status: Composing' indicator, 'Deploy', 'Stop', and 'Delete' buttons, and a 'View Graph' button. A 'Created' timestamp and an 'Owned by' field are also present. The browser's address bar shows the URL: http://10.6.24.116:9013/mario/app/default#eeae0533-5f11-4756-ae97-79dfd489aa15.

Currently selected tags: TableView

Facets automatically updated (Visualization facet and tags not compatible with TableView removed)

A graphical representation of the top ranking plan for the currently specified goal ("TableView")

MARIO creates ready-to-deploy application flows even for partially specified or ambiguous goals. In normal operation, business users can only focus on tags, and do not need to view flow graphs.



# MARIO User Interface

The screenshot displays the MARIO User Interface within a Mozilla Firefox browser window. The browser title is "Financial Services Flow Patterns : Unnamed (eeae) - Mozilla Firefox: IBM Edition". The address bar shows the URL: "http://10.6.24.116:9013/mario/app/default#eeae0533-5f11-4756-ae97-79dfd489aa15".

The main content area is titled "The IBM RESEARCH: Financial Services Flow Patterns" and features an "About" link. On the left sidebar, there are sections for "Click to create a new job:" with a "Create New" button, "Search existing jobs:" with a search input, "My Jobs" with a list containing "Unnamed (eeae)", and "Jobs Shared With Me" with a search input.

The central area shows a job configuration for "Unnamed (eeae) (click to rename)". It includes a "Please enter parameters:" section with the following fields:

- Industry: Tech
- TCP source host: @property:mario.source.csv.host@
- TCP source port: @property:mario.source.csv.port@

Below these fields are "Deploy Job with Parameter Values" and "Cancel" buttons. A callout box points to the "Deploy Job with Parameter Values" button with the text: "After clicking 'Deploy', the user can enter the parameters required by the generated application."

On the right side, there is a "Status: Composing" section with "Deploy", "Stop", and "Delete" buttons, a "View Graph" button, and a "Created: Monday, March 26, 2012 11:01:00 PM" timestamp. Below this are sections for "Owned by" and "Shared With", each with an "Add" button.

At the bottom of the interface, there is a status bar with "Active jobs: 0", a warning message "Warning: one or more server components failed to start.", and the footer text "Automated Analytics Composer - Web Application 5.2.2.20120301214638".

# MARIO User Interface

The screenshot displays the MARIO User Interface within a Mozilla Firefox browser window. The page title is "Financial Services Flow Patterns : ByIndustry TableView TcpSource (eeae) - Mozilla Firefox: IBM Edition". The URL is "http://10.6.24.116:9013/mario/app/default#eeae0533-5f11-4756-ae97-79dfd489aa15".

The main content area is titled "The IBM RESEARCH: Financial Services Flow Patterns" and contains a job titled "ByIndustry TableView TcpSource (eeae)". The job is active and has a status of "Active". It was created on Monday, March 26, 2012, at 11:01:00 PM. The job has several tags: BargainIndex, ByIndustry, ByTime, Complex, ExponentialIndex, OneCompany, Quote, TAQ, TableView, TcpSource, Trade, and VWAP. It is owned by the user and can be shared with others.

The data table shows the following columns: Time, VWAP, and BargainIndex. The data is streaming in, as indicated by the callout bubble.

Time	VWAP	BargainIndex
2012-03-26 23:30:41.515	35299.87274778814	0
2012-03-26 23:30:42.015	2302.262950894058	0
2012-03-26 23:30:42.815	10994.44882729349	1.817857967302705
2012-03-26 23:30:46.215	35266.17166662732	0
2012-03-26 23:30:47.141	35242.55217186235	75.01007716355807
2012-03-26 23:30:47.242	442.6909999080439	2.680788857372287
2012-03-26 23:30:46.415	9845.584028326125	79.11161100991800
2012-03-26 23:30:45.915	1910.904018778686	177.5621176164841
2012-03-26 23:30:43.115	9842.925929509359	0
2012-03-26 23:30:46.315	2305.406116803221	71.56711491688118
2012-03-26 23:30:47.343	2306.162244166880	0
2012-03-26 23:30:47.141	35242.55217186235	0
2012-03-26 23:30:47.343	2306.162244166880	0
2012-03-26 23:30:47.141	35242.55217186235	9869574347211
2012-03-26 23:30:44.815	942.8846611904206	0
2012-03-26 23:30:48.551	441.6288593704247	12.7292

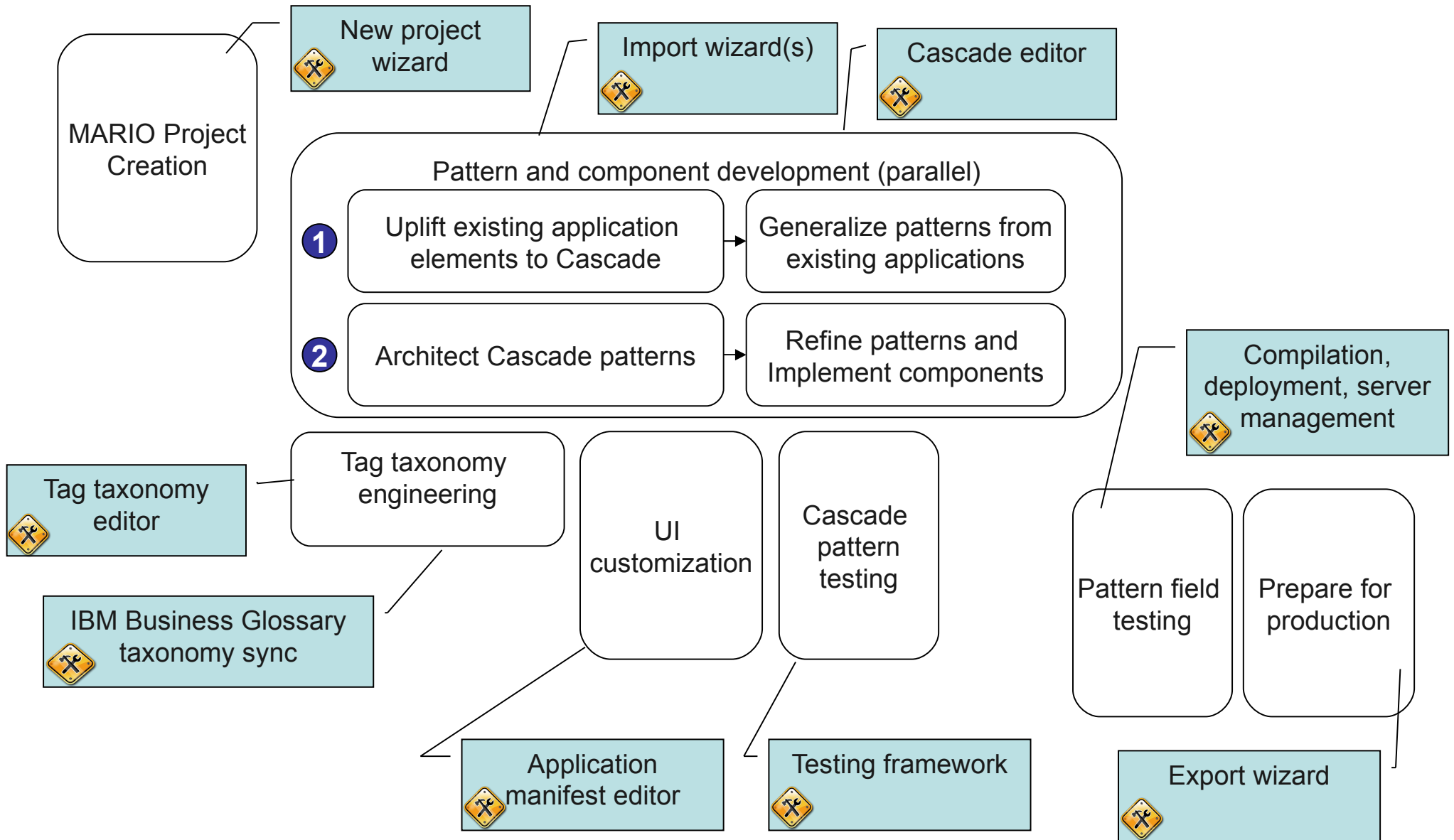
Active jobs: 1      Warning: one or more server components failed to start.

Transferring data from 10.6.24.116...

Version: 5.2.2.20120301214638

The results are streaming in.

# MARIO App Development Process and Tools



# See Also

- Our paper comparing HTN and Cascade
  - In SPARK workshop tomorrow
- MARIO Demo
  - System Demos -- 8pm Wednesday