ESPRIT Project N. 25 338

Work package I

Pilot Application 1

Software Status Report (DI4.1 – Part 1)

ID:	WP_I_SoftwareStatusReport	Date:	15.10.1998
Author(s):	Hans-Guenter Stein, FAST e.V.	Status:	deliverable
	Steffen Poellot, FAST e.V.		
Reviewer(s):		Distribution:	Project internal & EC reviewers

Change History

Document Code	Change Description	Author	Date
WP_I_SoftwareStatus Report	Version 0.1. No changes.	Stein, Poellot	15.08.98
WP_I_SoftwareStatus Report	Version 0.2. changes according to internal reviewers comments	Stein, Poellot	15.10.98

Overview

This document provides an intermediate progress report on software development in workpackage I which deals with the implementation of two pilot applications within the context of the Bavaria Online citizens network.

The document briefly outlines the user imposed requirements towards the applications. Next a high level application domain model is presented to outline the system's core components along with a roadmap for the physical deployment of these components. Moreover, a conceptual model is provided of how the application specific components are interfacing with other FollowMe components such as User Access, Agent Framework and Information Spaces. Furthermore a progress report on application implementation is provided. The document concludes with an outlook on ongoing and future activities within the implementation phase of workpackage I.

1. Application Requirements, Design and Implementation

In workpackage I a set of services is established, based on a framework that provides a generic method for constructing domain specific, user customizable services for context related information retrieval on the internet. The approach taken is to logically separate information sources (raw data) from services (data processing applications). The aim is to give end users high level access to customized information by providing them with access to services rather than letting them deal with locating and filtering raw data from potentially highly distributed information sources on the internet. This is referred to as the pattern of information latticework in the FollowMe architecture report [1].

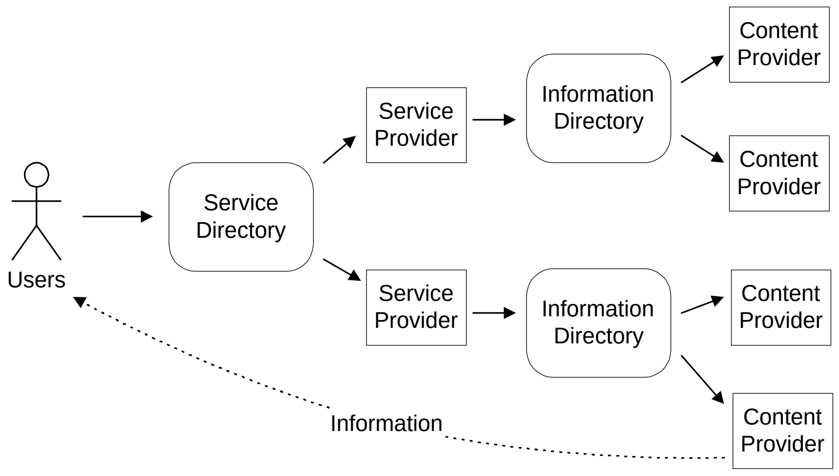

Figure 1: Information latticework

The framework is validated by two pilot applications implementing services in the domains of stock portfolio management and regional event notification. The applications are implemented in the Bavaria Online citizens network which provides a large base of potential users. System components will be deployed at FAST and five different Bavaria Online Citizens network nodes. The applications build on top of other FollowMe components (e.g. Mobile Object Workbench, Information Space, User Access, Agent Framework) thus validating the concepts of the FollowMe mobile agent architecture.

1.1 User requirements

The requirements towards the functionality offered by the pilot applications were derived from user and information provider surveys. The results of the surveys are documented in [3] and can be summarized as follows:

- provide domain specific, context related information retrieval facilities;
- information retrieval facilities should be user customizable and easy to access and use;
- provide components that, once personalized by the user, continuously monitor information sources on behalf of the user even if the user is not connected;
- provide means of transmitting information to the user via a variety of different output devices like fax, phone, e-mail, web-browser;
- provide means to schedule information delivery (triggered via alarms with recurrence rules)
- provide means to trigger information delivery due to external events (e.g. an information provider announces the availability of new or updated information);
- both suggested application domains (regional event notification and portfolio management) proved to be of interest for a majority of interviewed users.

These user and provider requirements have been analyzed and developed into use case scenarios (see requirements document of workpackage I [4]).

1.2 Application domains

Within workpackage I two pilot applications will be implemented and deployed within the context of the Bavaria Online citizens network.

1.2.1 Regional Event Notification System

Basic functionality

The Regional Event Notification System provides users with access to information on local events (e.g. cinema or theater events, local markets,...). This application supports the following core functionality (for details see [4]):

- Users may define their interests in specific events in form of query objects. The query specification basically holds the following information:
  - Information on the kind of regional events that are of interest to the user. Events are qualified by event type, by location of the event and by the timeline of the event.
  - Information on when the system should deliver notifications on upcoming events to the user (e.g. report every Friday at 17:00 on cinema events taking place over the next weekend).

- Data (information on local events) is collected from a set of distributed data sources (Information Providers) on a regular basis according to a user defined schedule.
- Periodic reports on the collected information are prepared and delivered (via Personal Assistant and User Profile) to the user by means of various output devices such as fax, e-mail or SMS.

Application domain model

Figure 2 outlines the top level classes used to implement the Regional Event Notification System.

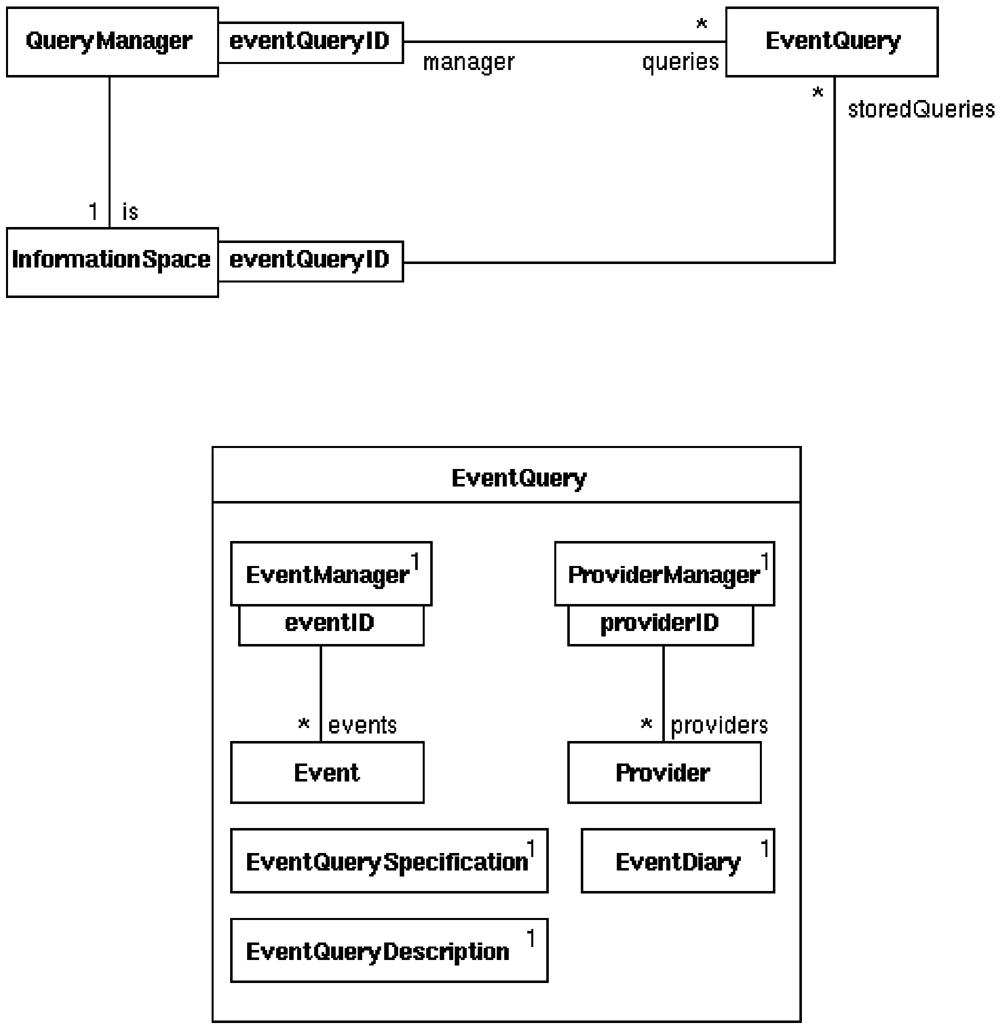

Figure 2: Top level class diagram for Event Notification System

Each individual user customized query along with the results returned by that query are encapsulated in EventQuery objects. The EventQuery objects are maintained by a QueryManager which persistently stores these objects in the user’s Information Space. User specified query parameters are stored in an EventQuerySpecification. Query execution times and report delivery schedules are stored in an EventDiary object. Query results retrieved from information providers are stored as Event objects. The list of Event objects is maintained by an EventManager object. A ProviderManager object maintains a list of Provider objects which store information on information providers.

User interfaces for configuration of queries and for presentation of query results are generated by the individual EventQuery objects. Each data object encapsulated in an EventQuery object can output its data contents in form of an XML string. These XML strings along with static XSL strings defining the layout of the data are handed over to the User Access components to render the user interfaces on appropriate output devices.

More details on design issues can be reviewed in [4] and [6].

Physical deployment of components

In the event notification domain information is provided by local event organizers (i.e. cinemas, schools, etc.). They communicate their event advertisements to the *Bavaria Online* nodes by means defined by the individual *Bavaria Online* nodes themselves. The event advertisements are stored in databases. These databases along with proper interfaces to the clients executing the queries (agents) are hosted by each participating Bavaria Online node.

All user specific components are also hosted by the Bavaria Online nodes (e.g. Personal Assistants, Task Agents, Information Spaces, Personal Profiles – see below: 1.3).

Each Bavaria Online node hosts its own modules to facilitate communication with the users (User Access components – see below: 1.3).

FAST hosts traders for locating all kind of objects within the system (e.g. Information Providers, Personal Assistants, Service Providers – see below: 1.3).

FAST also hosts a third party geographical information system used to pin down locations of regional events.

1.2.2 Stock Portfolio Management System

The Stock Portfolio Management System allows users to maintain their personal stock portfolios. Share values are automatically and continuously updated by querying standard share value information providers like Yahoo! Finance. The application supports the following core functionality (for details see [4]):

- Users may maintain an arbitrary number of stock portfolios. Portfolio objects hold information on shares owned by the user, dates of share transactions, share values, transaction histories, etc.
- Portfolio objects register with one or more share value servers. Share values are continuously updated (e.g. every 5 minutes).
- Users may specify a schedule for periodically receiving reports on the current value of their portfolios. Reports may be delivered via a variety of output devices (e.g. fax, e-mail, SMS).
- Users may attach limits to the values of the shares they own. Whenever limits are exceeded, the system will trigger instant notification of the user. Notifications may be delivered via a variety of output devices (e.g. fax, e-mail, SMS).

Application domain model

Figures 3-5 outline the top level classes used to implement the Stock Portfolio Management System.

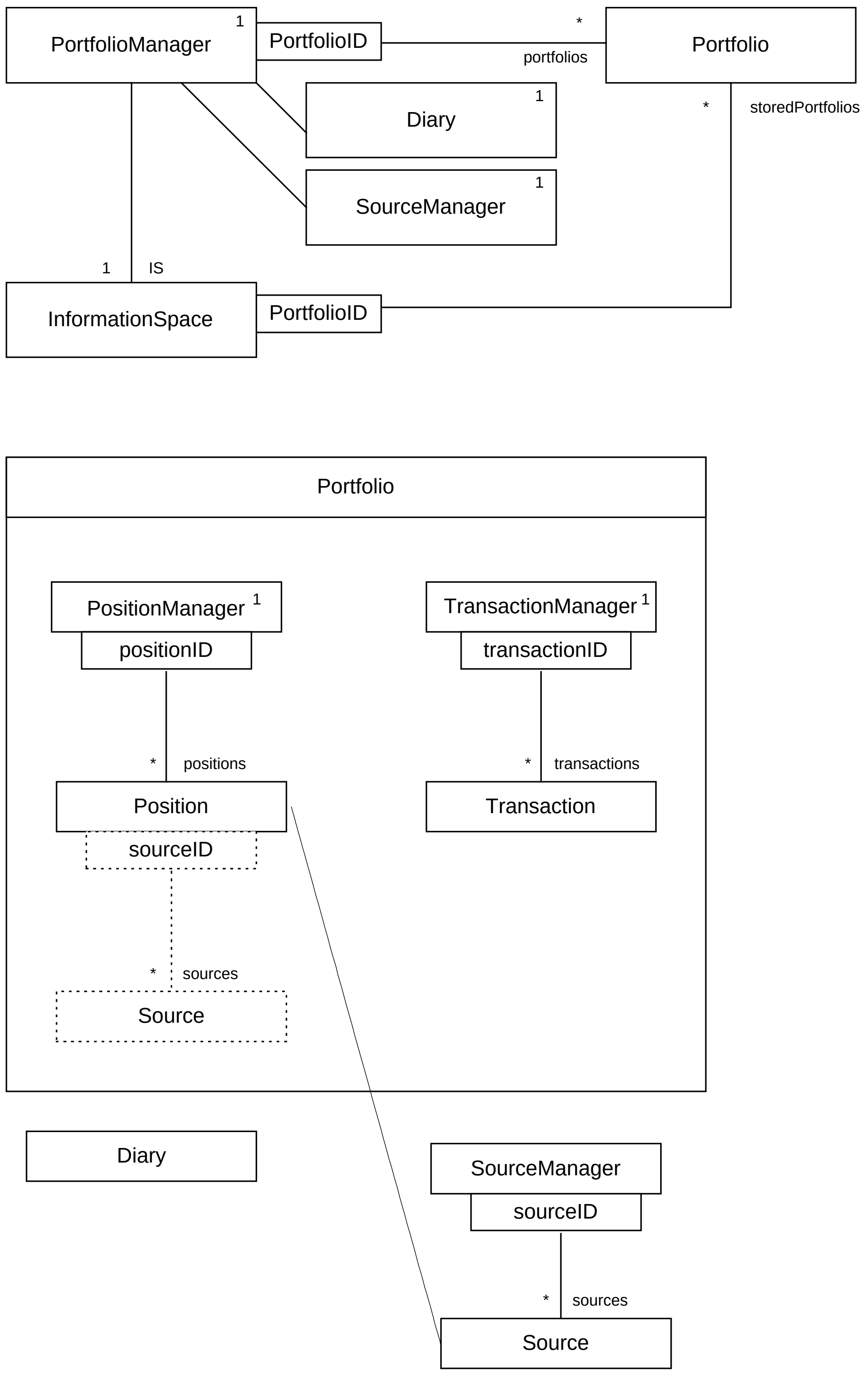

Figure 3: Top level class diagram for user specific components

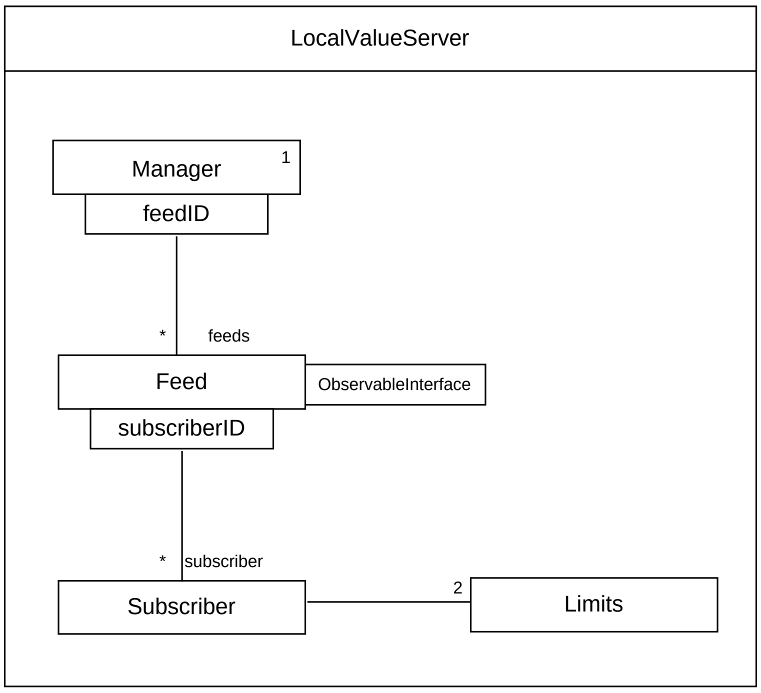

Figure 4: Top level class diagram for local value servers

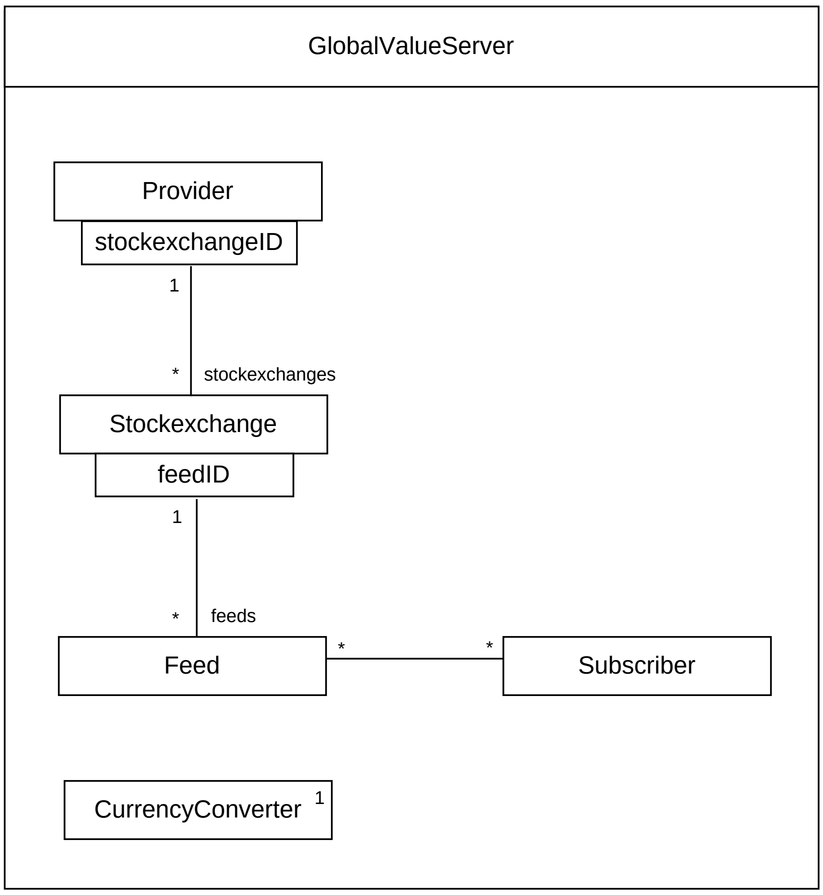

Figure 5: Top level class diagram for the central value server

Each individual user specific stock portfolio is encapsulated in Portfolio objects. The Portfolio objects are maintained by a PortfolioManager which persistently stores these objects in the user's Information Space. The PortfolioManager also maintains a Diary object and a SourceManager object. The Diary object holds information on report delivery schedules. The SourceManager holds information on the data sources that are used to retrieve share values. A Portfolio holds information on stock share positions encapsulated in Position objects which are managed by a PositionManager. In addition each Portfolio tracks all transactions (e.g. stock share purchases). This information is encapsulated in Transaction objects which are managed by a TransactionManager. Stock Position objects hold information on share values which are retrieved from certain data sources as outlined in the physical deployment diagram in figure 7. Information on share value servers is maintained by a SourceManager object.

Portfolio objects receive share values from local value servers by subscribing as listeners to share value feeds. Local value servers are share value servers hosted on the same host than the Portfolio objects. They collect requests for specific share values and thus serve as proxy value servers. The local value servers in turn subscribe to share value feeds from a central value server which actually connects to well known system external share value servers like Yahoo! Finance US and Yahoo! Finance Europe. The PortfolioManager object may directly query these share value servers for ticker symbol lookups as outlined in figure 7.

User interfaces for creation and maintenance of portfolios (e.g. stock share transactions), for presentation of scheduled reports on current portfolio contents and values and for instant notification on share values exceeding user defined limits are prepared by the individual Portfolio objects. Each data object encapsulated in an Portfolio object can output its data contents in form of XML strings. These XML strings along with static XSL strings defining the layout of the data are handed over to the User Access components to render the user interfaces on appropriate output devices.

More details on design issues can be reviewed in [4] and [6].

Physical deployment of components

Physical deployment of components within the Stock Portfolio Management System is outlined in Figure 6.

The individual Bavaria Online nodes host local value servers as proxies for share value request from individual users as well as the user specific components themselves.

In addition, each Bavaria Online node hosts its own modules to facilitate communication with the users (User Access components – see below: 1.3).

The central value server interfacing with the external share value providers is hosted at FAST.

FAST also hosts traders for locating all kind of objects within the system (e.g. Personal Assistants – see below: 1.3).

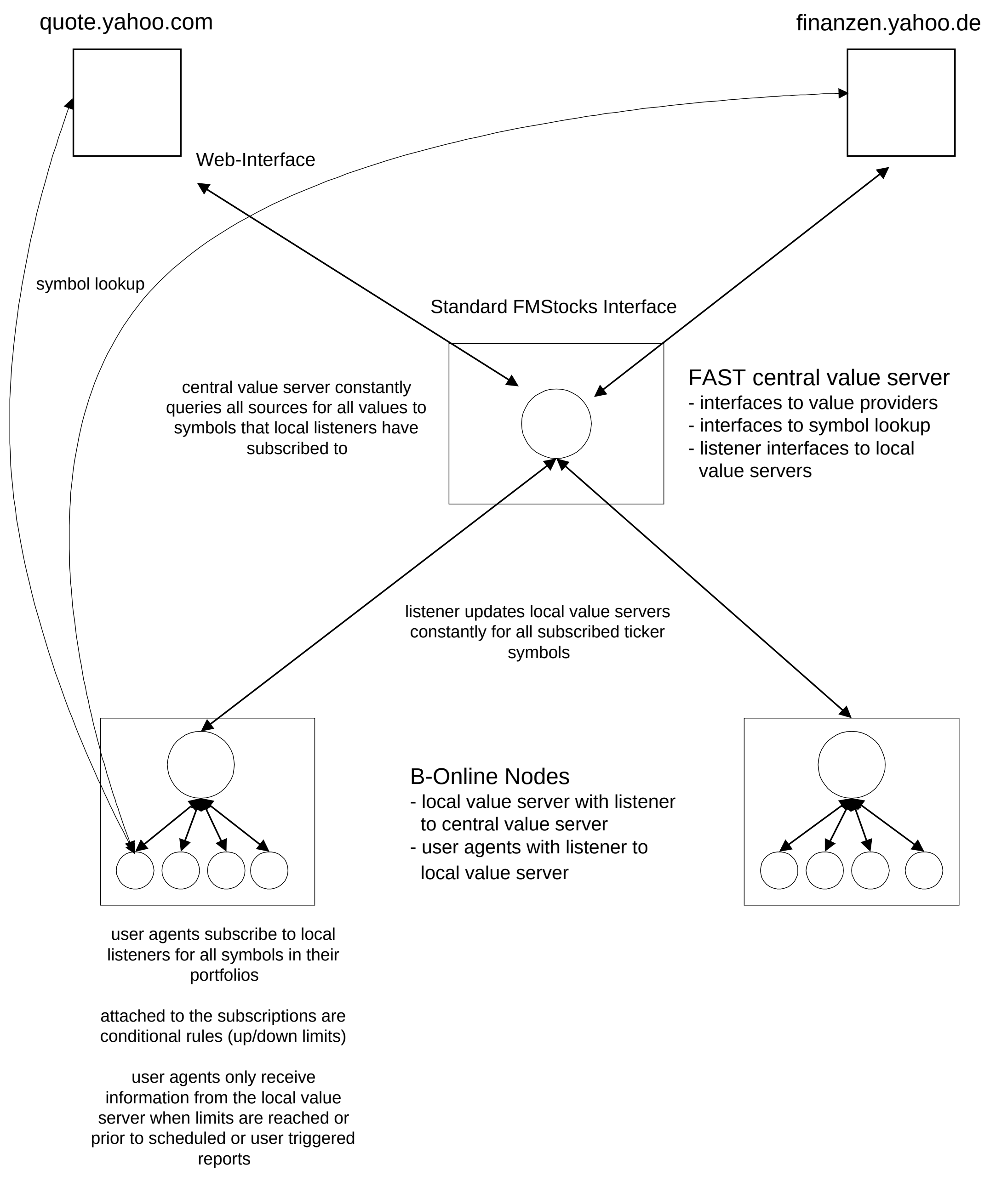
quote.yahoo.com
finanzen.yahoo.de
Web-Interface
symbol lookup
Standard FMStocks Interface
central value server constantly queries all sources for all values to symbols that local listeners have subscribed to
FAST central value server
- interfaces to value providers
- interfaces to symbol lookup
- listener interfaces to local value servers
listener updates local value servers constantly for all subscribed ticker symbols
B-Online Nodes
- local value server with listener to central value server
- user agents with listener to local value server
user agents subscribe to local listeners for all symbols in their portfolios
attached to the subscriptions are conditional rules (up/down limits)
user agents only receive information from the local value server when limits are reached or prior to scheduled or user triggered reports

Figure 6: Physical deployment of system components

1.3 Integration with the FollowMe Framework

Both workpackage I pilot applications integrate into the FollowMe Agent Framework [2] as outlined in figure 2. The Agent Framework acts as an agent-based infrastructure for the applications in the sense that it provides:

- Storage management facilities: The Personal Assistant manages access to a logical object storage facility (the Information Space) so that applications can persistently store and recover any Java objects.

- User manager: The Agent Framework provides authentication and security facilities to ensure that every user only has access to her own components (Personal Assistant, Information Space, Task Agents).
- User interfacing facilities: The Personal Assistant provides mechanisms to communicate with the user via the User Access components.
- User preferences management facilities: The Personal Assistant provides access to user specific data stored in Personal Profiles (e.g. user name, address, phone numbers,...).
- Trigger management facilities: The Agent Framework provides a diary manager with means to trigger actions by alarm settings including recurrence rules.
- Trading facilities to enable applications to locate service interfaces by type or service properties.

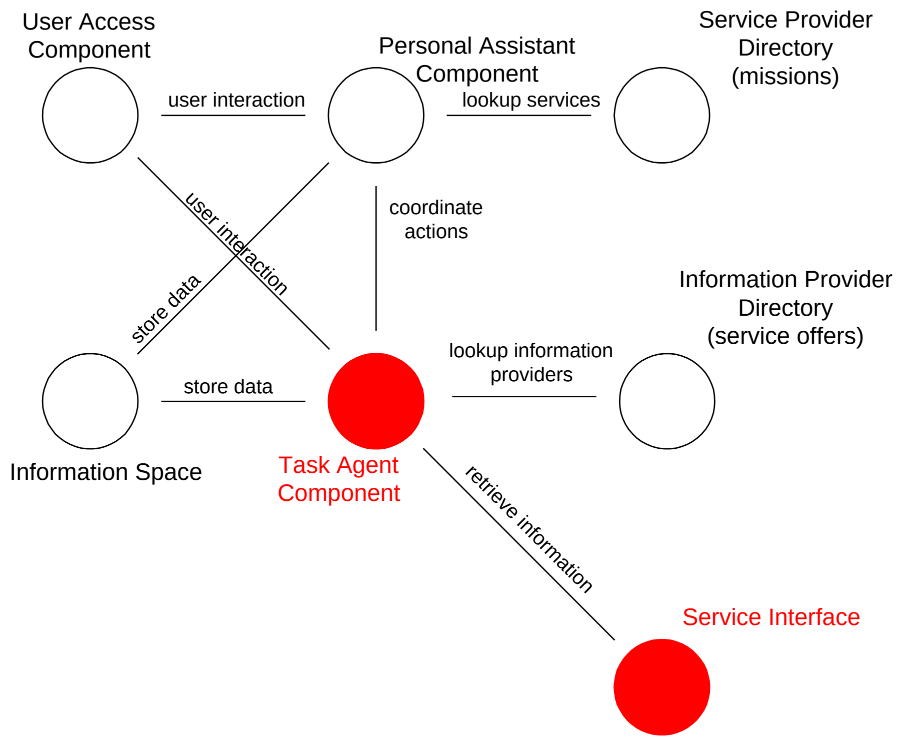

Figure 7: Domain model

Access to application specific methods is controlled by a top level control object (Task Agent) that is implemented as a Mission object. Missions are registered at the Agent Trader and can be launched via the Personal Assistant. Service interfaces to information provider sites are registered as service offers at the Agent Trader and can be accessed by the Task Agent. The Task Agent controls a set of application specific objects and stores persistent data in an Information Space [5]. Access to the Information Space is controlled by the Personal Assistant. As outlined in [2], every user may store personal data (e.g. user name, phone number,...) in a Personal Profile. The Task Agent has access to this data via the Personal Assistant. User interaction is mediated by the User Access components. The Task Agent may directly interact with User Access or use the Personal Assistant's *send* method to do so.

2. Status of the Implementation

2.1 Current status

The time schedule for development of FollowMe Agent Framework components overlaps with the timeline for the implementation of the pilot applications. For that reason the implementation of pilot applications was divided into two phases:

1. Implementation of application specific Java classes
2. Integration with the FollowMe Agent Framework

However, to facilitate testing of the implementation of application specific Java classes at least a rudimentary integration of object storage facilities using Information Space components and User Access components was required. These implementations use interfaces that are consistent with Agent Framework API specifications as outlined in [8].

2.1.1 Regional Event Notification System

A first prototype of the Regional Event Notification System has been implemented (see [9] and [10] for source code). The prototype consists of user specific classes along with appropriate user interfaces in form of XML and XSL documents, two example SQL databases implemented in Microsoft Access which hold faked data on regional events and a third party geographical information system providing graphical representations of event locations.

The prototype currently builds on User Access version 1.1 and MOW version 2.0 (including Information Spaces). Agent Framework components are not integrated in this first release.

Database interface specifications and data models have been completed in co-operation with the Bavaria Online node operators (information providers). Bavaria Online node operators started to build up their own databases holding real information on regional events.

Technicians at Bavaria Online nodes have access to the prototype currently hosted at FAST.

Instances of the application (task agents) are manually created for every new user. Task agents are not protected by authentication methods or other security features.

Once instantiated, task agents can be used to define multiple query objects. The queries are executed against the two example databases distributed among the FAST internal network. The interfaces to these databases are registered as service offers at a trader. The query objects locate these service offers at runtime so additional databases could be introduced just by add the respective service offer to the trader. Reports on executed queries can be sent to e-mail, fax or SMS devices or can be viewed online using a web-browser.

API specifications generated by JavaDoc can be reviewed at [11]. The content of the Java packages can be summarized as follows:

- **DE.fast.followme.pilotapplication**: core control classes for Personal Assistant and Task Agent
- **DE.fast.followme.pilotapplication.event**: application specific classes of the event notification system
- **DE.fast.followme.pilotapplication.utilities**: supporting utilities like Personal Assistant admin interface, database trader, JDBC connectors, Java reflection modules, etc.

2.1.2 Stock Portfolio Management System

The additional effort required to provide rudimentary implementations of infrastructure components (e.g. communication with User Access, maintenance of Personal Profiles and Diaries, access to the Information Space,...) and later migration to the infrastructure provided by the Agent Framework proved to be quite substantial. Therefore implementation of the second Bavarian pilot application, the Stock Portfolio Management System, was delayed until first versions of the complete Agent Framework become available. Effort spent on implementation was therefore reduced during project month 8-11 and will rise again during project month 12-15.

2.2 Next steps

In the following sections we briefly outline next steps in software implementation and deployment.

2.2.1 Regional Event Notification System

The next steps in the implementation process of the Regional Event Notification System are to integrate the existing classes with Agent Framework release 1.2 and User Access release 1.2.

Once this is completed all components will be deployed at the Bavaria Online nodes for public access by Bavaria Online users. Evaluation of the usage of the application will start after deployment.

Integration with Agent Framework 1.2 [2] and User Access 1.2 involves [7]:

- The top level control class of the application (namely the task agent) will be converted into a mission script object using the Agent Framework scripting facilities.
- For scheduling of alarms UWE’s diary alarm functionality will be used. Alarms no longer trigger methods in Java classes but trigger functions declared in the mission script. The mission object will be added to the alarm object as a listener.

- All trading will be done using the Agent Framework's AgentTrader (e.g. trading of database interfaces, User Access locations, locating of a user's own Personal Assistant,...).
- The Agent Framework's Personal Assistant components will be integrated so that task agents (mission objects) can be launched using a Personal Assistant.
- Upgrading to User Access release 1.2 requires to restructure sending of documents to the user and handling of user feedback (e.g. via forms). The current prototype uses a proprietary event handler. Within version 1.2 of User Access event handling is encapsulated within the documents sent to the user via a User Access connection.
- Initialization of communications with the user via User Access connection objects will be shifted from the Task Agent to the Personal Assistant.

The currently available Agent Framework 1.2 does not provide locating mechanisms for active Personal Assistants in a distributed environment via a trader. It also does not provide security and authentication facilities. These will be available with release 1.3.

Deployment of components at Bavaria Online nodes will progress as follows:

- Java database connectors (interfaces) will be deployed at all Bavaria Online nodes. These connectors will be registered at the database trader at FAST so that task agents and there query objects can locate them and start querying the attached databases.
- All technicians at Bavaria Online nodes will have there own Personal Assistants set up at FAST. Via these Personal Assistants they can launch their own Task Agents and access their individual Information Spaces.
- After Bavaria Online technicians tested agents running at FAST, Agent Factories will be installed at Bavaria Online nodes and the application will be announced among the Bavaria Online user community.
- User Access components will be set up at Bavaria Online nodes.
- Access logging mechanisms will be incorporated into the agents to facilitate the evaluation process.

2.2.2 Stock Portfolio Management System

Implementation of the Stock Portfolio Management System will be started after successful integration of the Regional Event Notification System with the Agent Framework and the latest User Access release. That way the implementation can build on latest releases of FollowMe middleware components thus avoiding overhead in workaround implementations of components belonging in other workpackages. Making use of experiences gained during the process of integrating the Regional Event Notification System prototype with the latest releases of Agent Framework and User Access will also speed up the implementation process.

Deployment and evaluation of Stock Portfolio Management System components will proceed in analogy to the deployment and evaluation process of the Regional Event Notification System as outlined above.

3. References

[1] FollowMe, APM Architecture Report, Deliverable DA1.2

[2] FollowMe, UWE Agent Framework Guide version 1.02

[3] FollowMe, FAST Requirements (Supplement), Deliverable DI2.a

[4] FollowMe, FAST Requirements and Architectural Design, Deliverable DI2

[5] FollowMe, APM Information Space Design, Deliverable DC2

[6] FollowMe, FAST Design, Deliverable DI3

[7] FollowMe, FAST User Access Software Report, Deliverable DH6.3 (Draft)

[8] FollowMe, UWE Autonomous Agents Design, Deliverable DD3

[9] FollowMe, FAST Pilot Prototype Source Code, http://hyperwave.fast.de/FASTPAv01

[10] FAST FAST Pilot Prototype Source Code Installation Guide, http://hyperwave.fast.de/FASTPAv01.doc

[11] FollowMe, FAST Pilot Application 1 Java API Documentation, http://hyperwave.fast.de/FASTPA_1_JavaDoc_API