Harnessing Events

ICS221 Software Engineering
Winter Quarter 2006

Roberto Silveira Silva Filho
rsilvafi@ics.uci.edu

Jan 31st, 2006.

Outline

- Events and Software Engineering
  - Background
  - Publish/subscribe interaction
- Applications
  - Expectation-Driven Monitoring (EDEM)
  - Knowledge Depot
  - CASSIUS (Awareness)
- Infrastructures
  - Event Notification Services
    - SIENA (Content-based network)
  - Current Research
    - YANCEES
- Conclusions

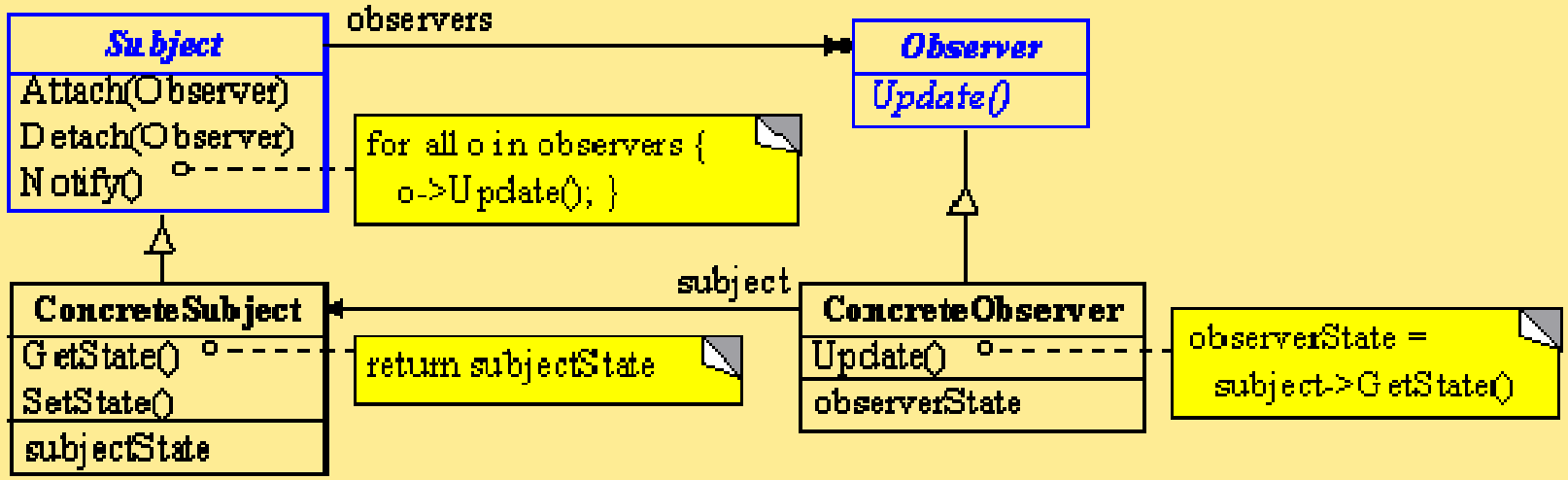
Events

- An *event* expresses:
  - a state change (or transition) in a software system component
  - or represents some temporal fact in the real world

- Both the world and computational systems are in constant transition, generating and consuming information.
- Events are computational representations of those temporal transitions and state changes.
- Event-driven applications respond to those computational and environmental changes.
- They are usually build according to the publish/subscribe communication style or pattern.

Publish/subscribe interaction

- Also known as implicit invocation
- Implemented by the Observer design pattern [Gamma et al. 95]



Publish/subscribe main characteristics

- One-to-many communication
- Loose coupling with respect to:
  - Time (asynchronous notifications)
  - Location (possibly distribution)
- Anonymity
  - publishers (subjects) and subscribers (observers) do not need to know about each other

Publish/subscribe benefits

- Support for system evolution
  - publishers and subscribers can be easily added/removed
  - Subsystem independence
- Dynamic change
  - Potential for fault tolerance
- Heterogeneous systems integration
- Multicast communication is inherent to the model

Applications

- Software Engineering
  - User Interface Analysis and Design (EDEM [HR98])
  - Testing (MONARC [Dias2004])
  - Architectural Styles (C2 [TMA+96])
  - User interface toolkits (AWT for eg.)
- Awareness
  - Organizational (Knowledge Depot [KRZ97])
  - Integration (CASSIUS [KR01])
  - Presence (Portholes [GLT99])
  - Security and collaboration (Impromptu [DePaula+05])
  - Software Development Workplace (Palantír [SNH03])
- Workflow Management Systems (ex. JEDI [CN01])
- System integration
- And many others…

Event based applications

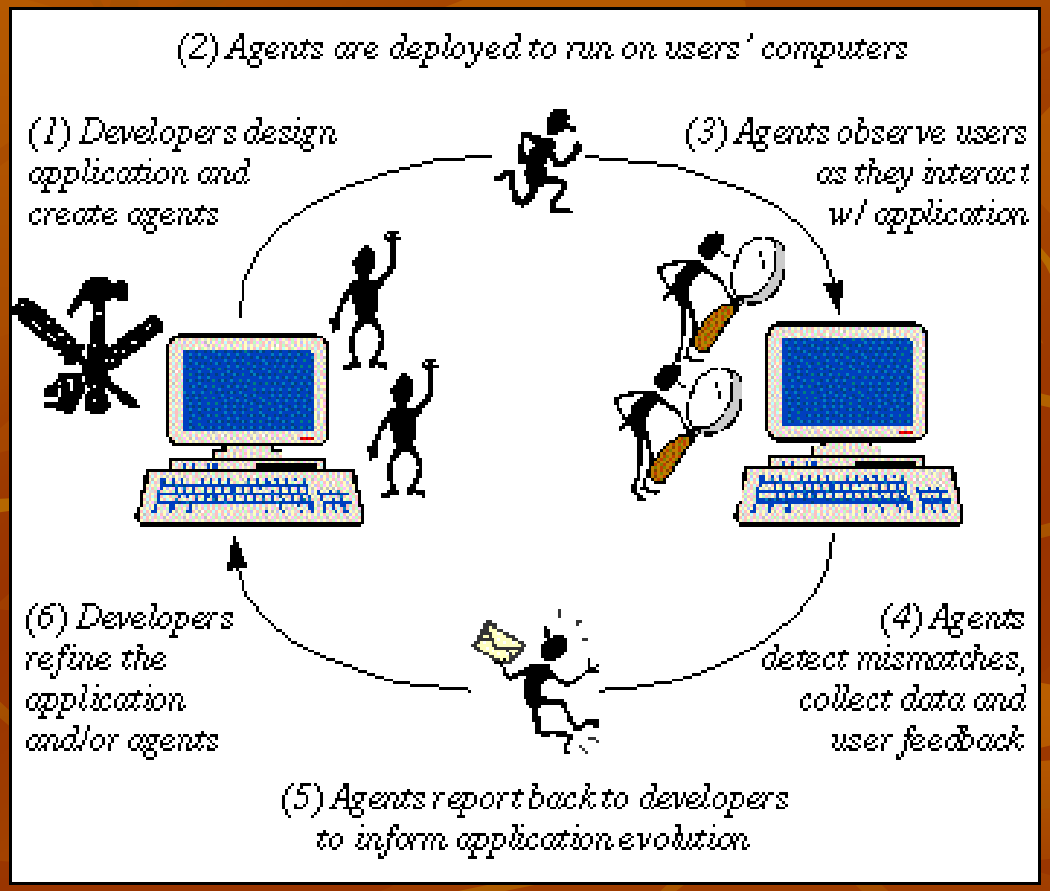
- EDEM
- Knowledge Depot
- CASSIUS

Background

- Software Design Expectations
  - Expectations influence designs, designs embody expectations
  - Mismatches between expectations and how applications are actually used can lead to breakdowns.
  - Identification and resolution of mismatches can help improve fit between design and use
  - Identifying mismatches entails observing actual use and comparing it against expectations.
- Expectation-Driven Event Monitoring
  - techniques to enable large-scale incorporation of usage data and user feedback in SW development, to help uncover mismatches and improve the design-use fit.

Approach

- Expectation-Driven Event Monitoring (EDEM)

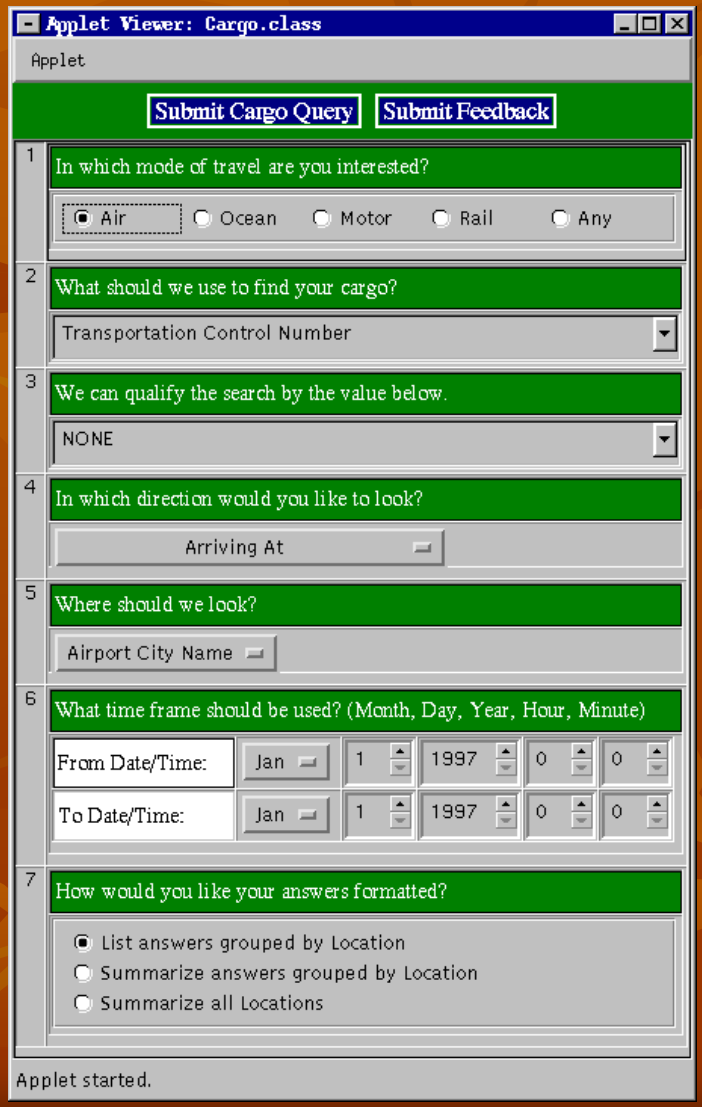


Approach

- Developers
  - design applications and identify usage expectations
  - create agents to collect usage data and user feedback
- Agents
  - deployed over the Internet to run on user computers (via HTTP)
  - perform abstraction, selection, reduction, context-capture as needed to allow actual use to be compared against expectations
  - report data and feedback to developers (via E-mail)
- Data and feedback
  - inform further evolution of expectations, application, and agents

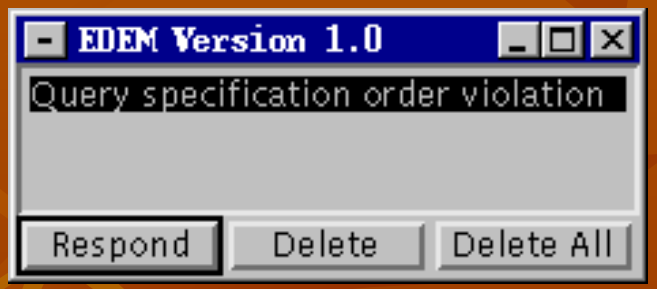
Usage scenario

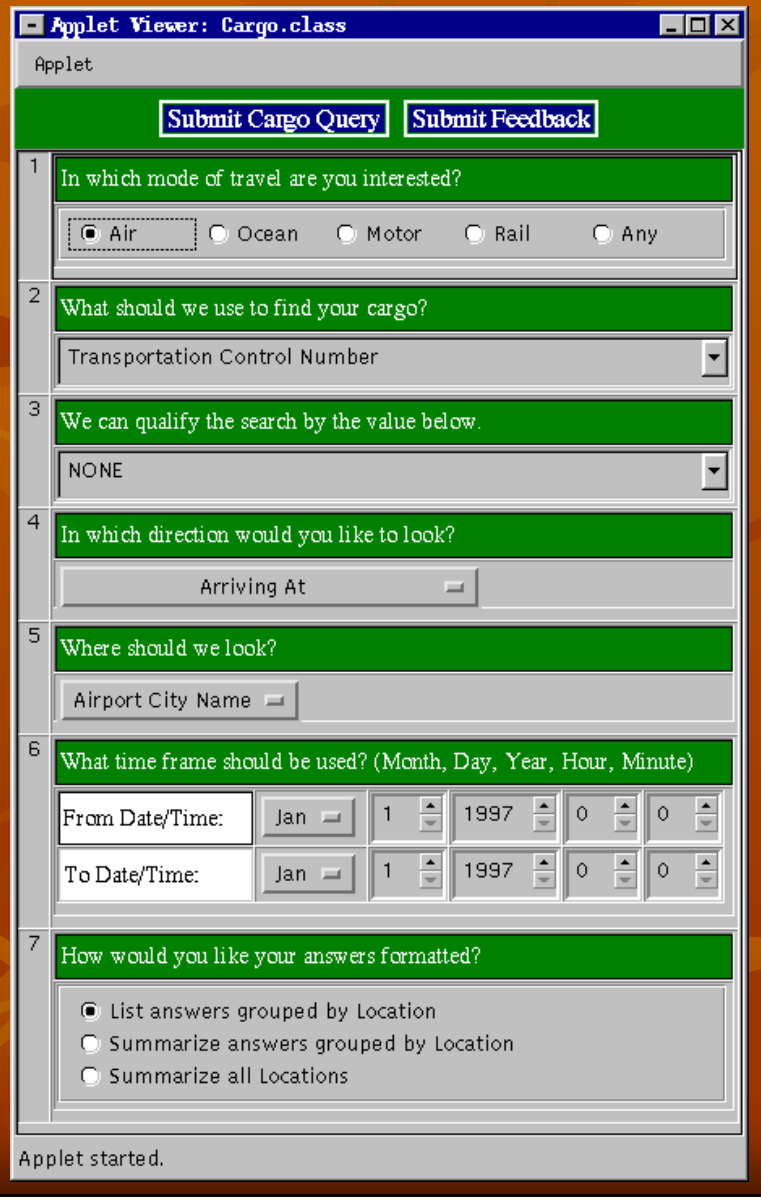
- A cargo query form



Agents

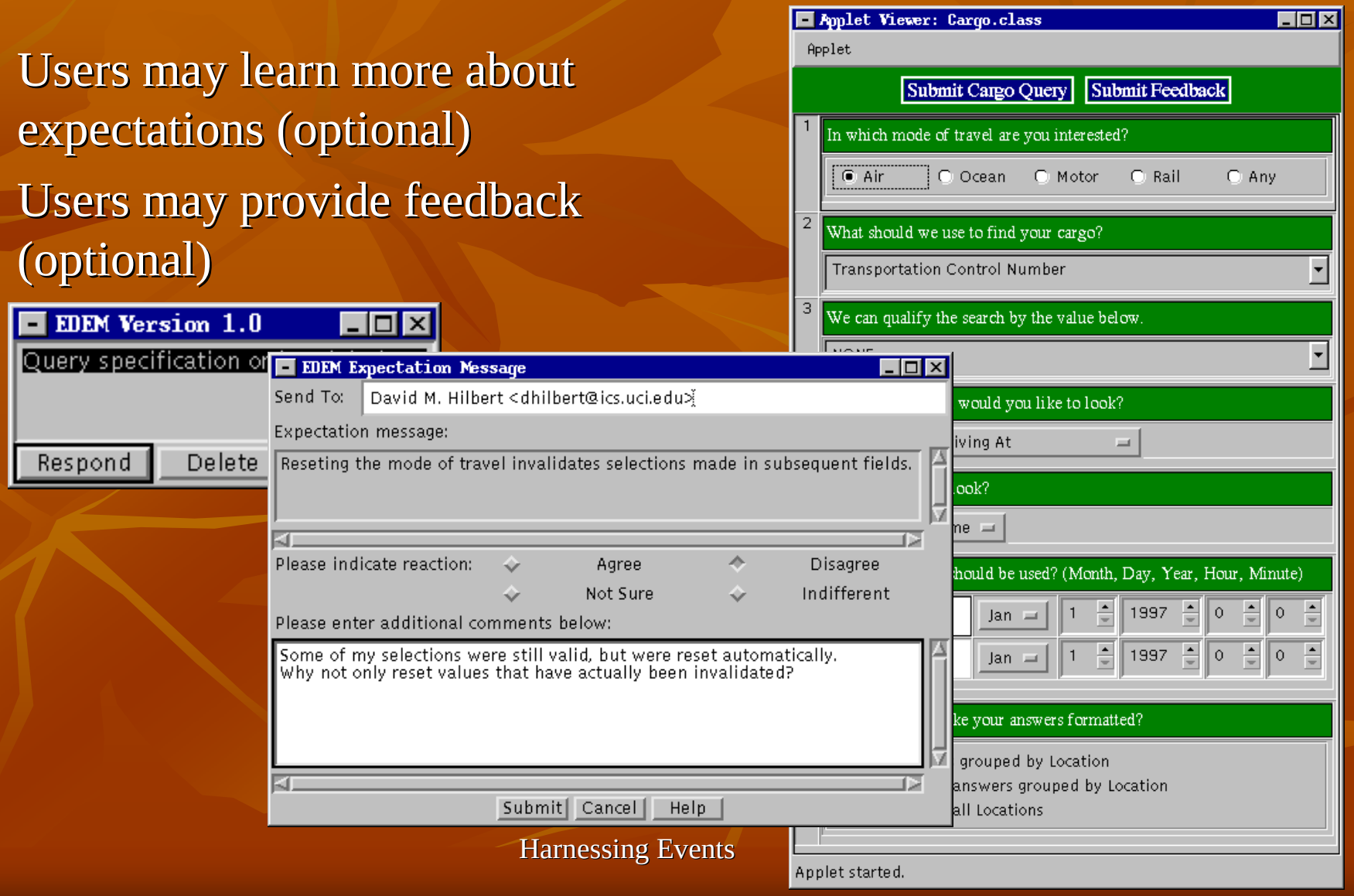
- Agents monitor use and collect data unobtrusively
- Agents may post messages (optional)





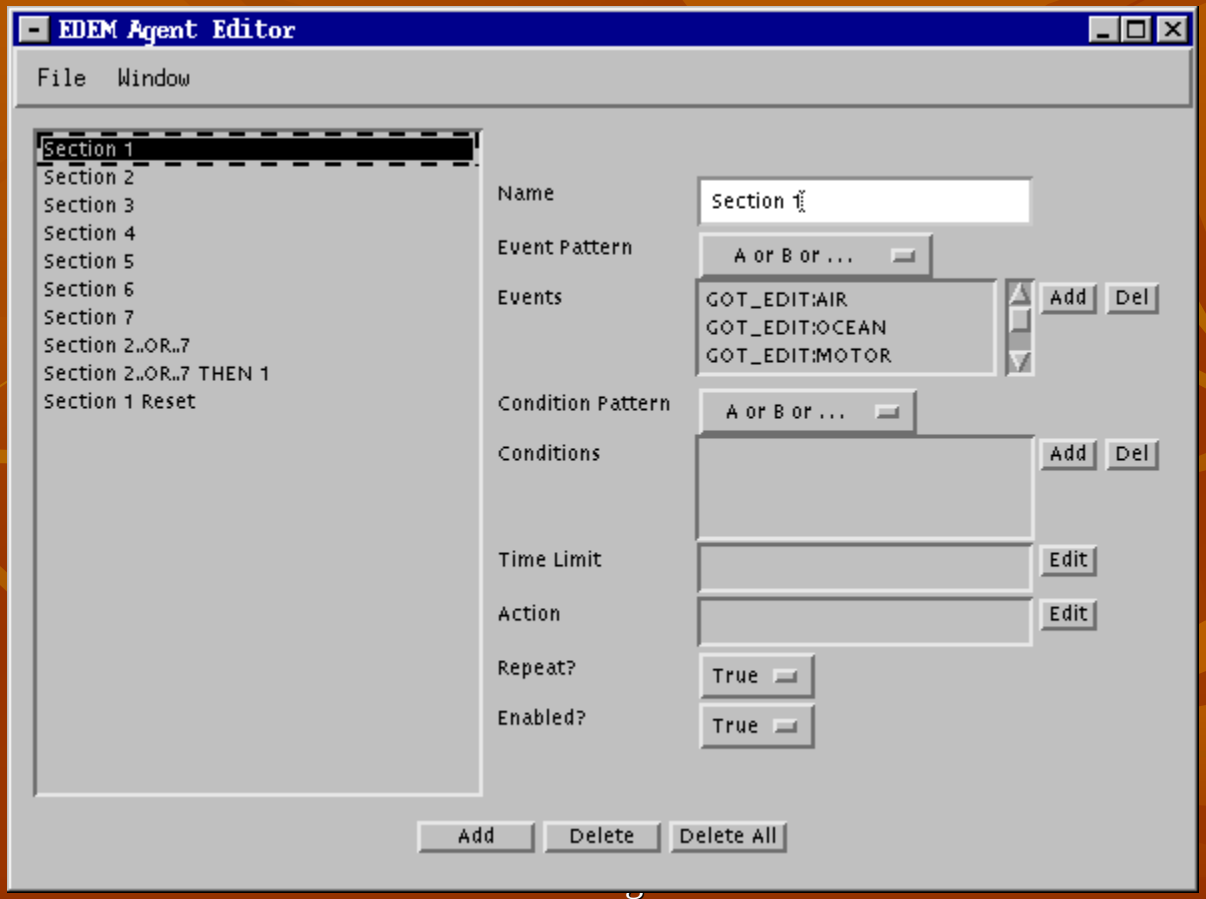
Users

- Users may learn more about expectations (optional)
- Users may provide feedback (optional)



Agent Authoring

- An agent that fires when the “mode of travel” section is edited



Agents

Trigger

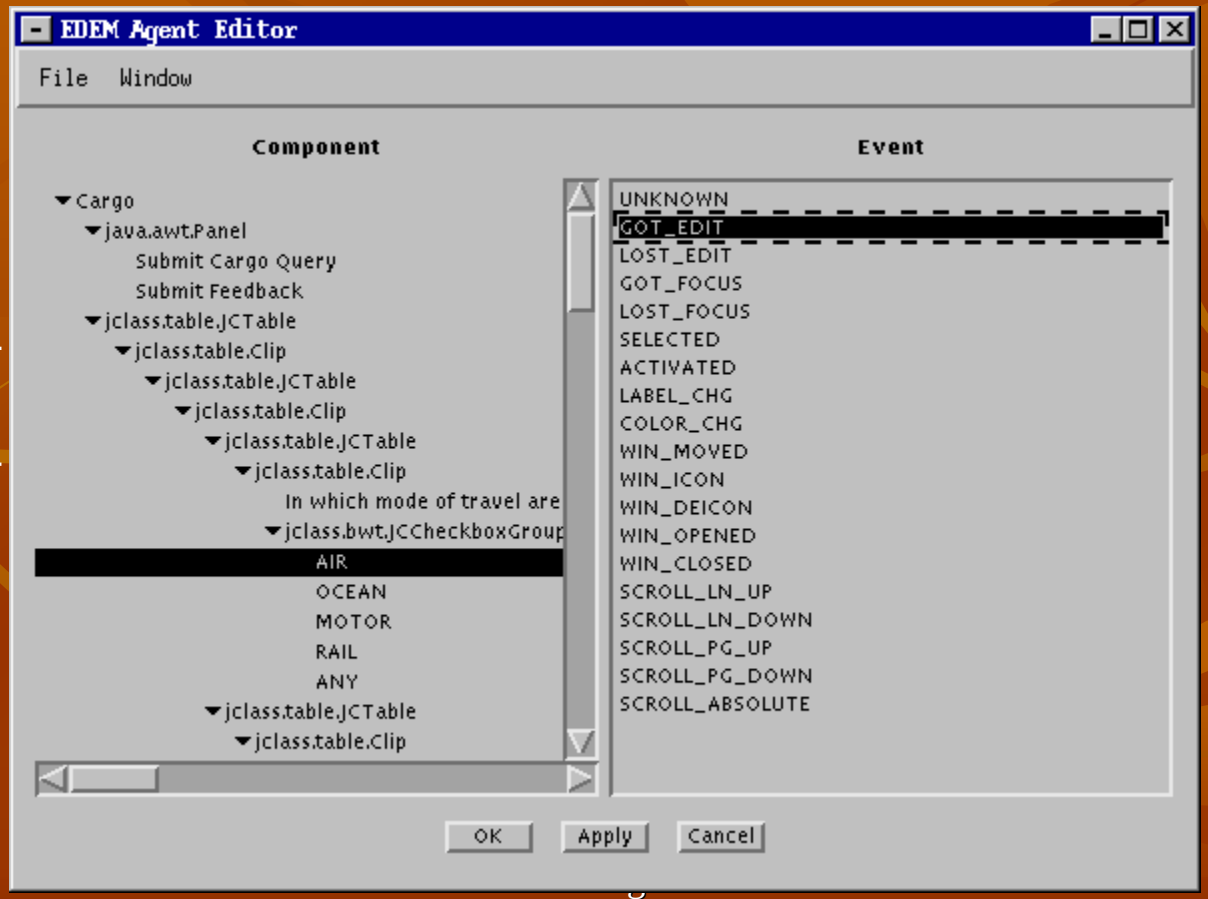
Guard

Actions

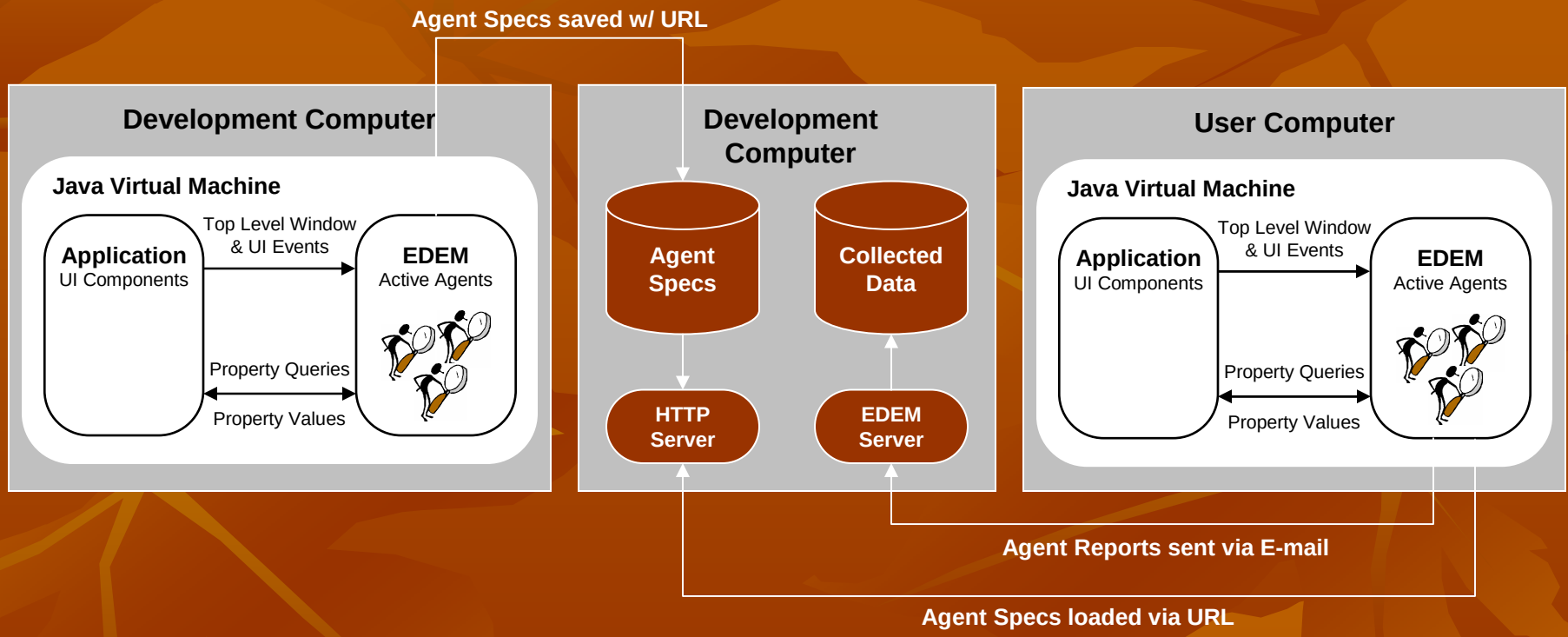
Event Specification

- Detecting when the user selects “AIR” as the “mode of travel”

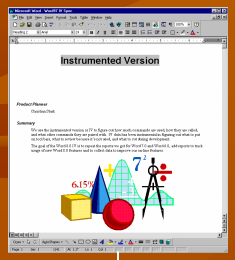
Widgets { } Events



Architecture



Reference Architecture (see ACM Computing Surveys)



Data Capture

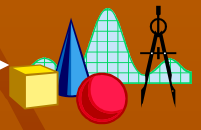
Abstraction, Selection, Context, Reduction

Data Packaging

Data Transport

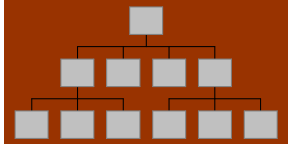
Data Prep

Data Analysis



System Model of UI & App:

Components
Events
Properties
Methods

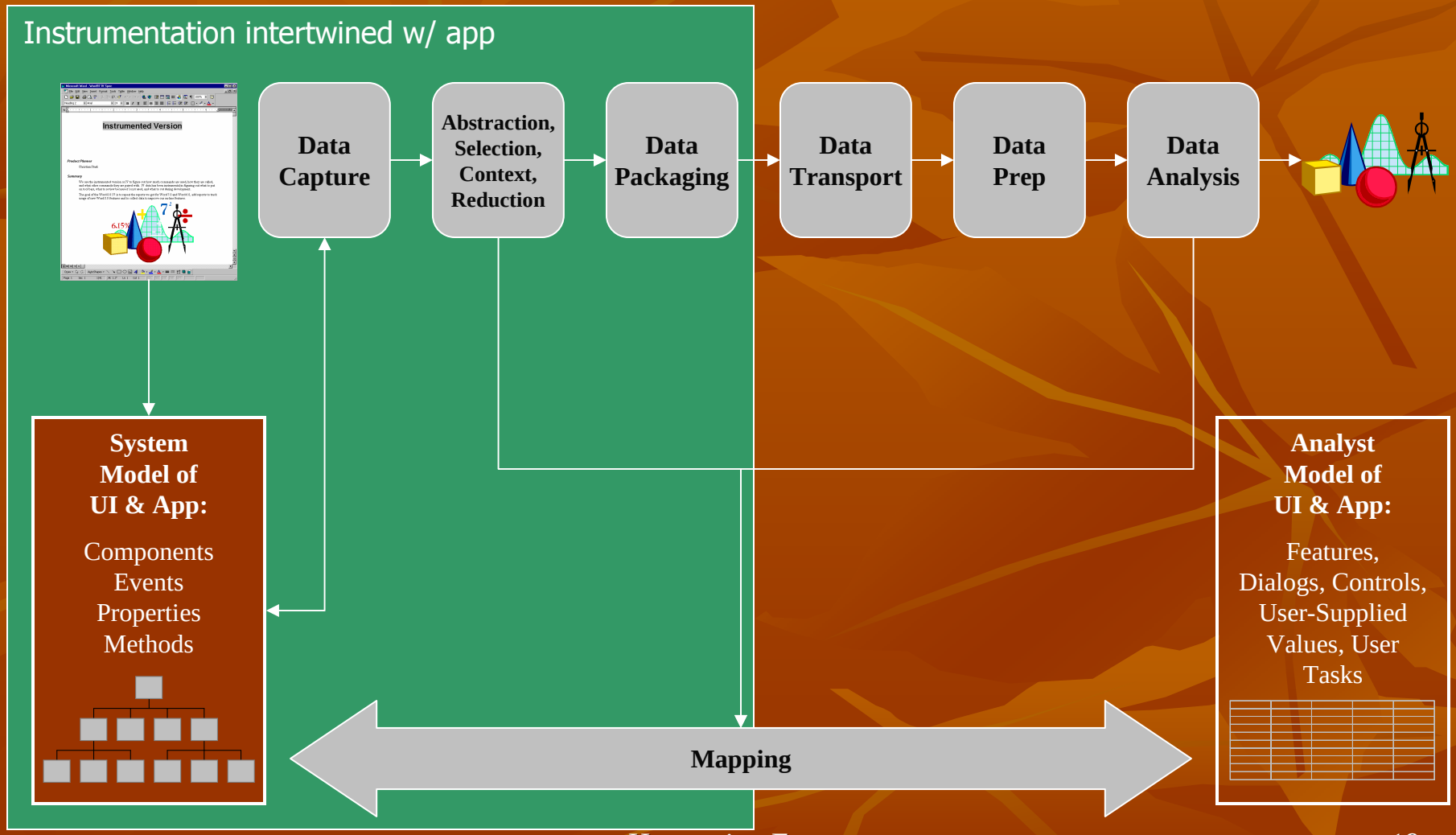


Mapping

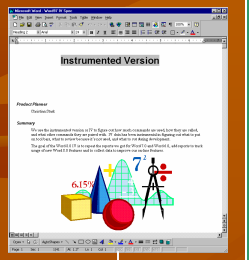
Analyst Model of UI & App:

Features, Dialogs, Controls, User-Supplied Values, User Tasks

Reference Architecture
Traditional Instrumentation



Reference Architecture (EDEM)



Event monitoring infrastructure

Data Capture

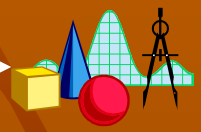
Abstraction, Selection, Context, Reduction

Data Packaging

Data Transport

Data Prep

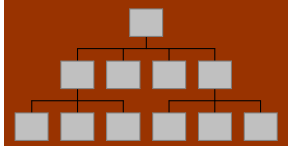
Data Analysis



"Pluggable" Data Abstraction, Selection, Context-Capture, and Reduction

System Model of UI & App:

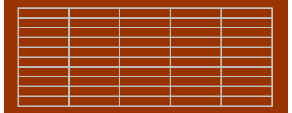
Components
Events
Properties
Methods



Mapping

Analyst Model of UI & App:

Features,
Dialogs, Controls,
User-Supplied Values, User Tasks



Expectation Agents

EDEM - Conclusions

- Usage expectations
  - help guide data collection
  - raise awareness of implications of design decisions
- Agent architecture
  - abstraction, selection, reduction in-context and prior to reporting
  - independent evolution of instrumentation and application
- Combined
  - higher quality data (v. beta tests) with less restrictions on evaluation size, scope, location, duration (v. usability tests)
  - a complementary source of usage and usability information

Other Possible Applications

- Use of long-term information about user and users' behavior to support
  - adaptive UI and application behavior
  - "smarter" delivery of help/suggestions/assistance
- Support for monitoring of other software systems that demands
  - Instrumentation: event and state information can be easily "tapped"
  - Abstraction: low-level data must be related to higher level concepts of interest
  - Scale: available information exceeds that which can practically be collected
  - Evolution: data collection needs evolve over time more quickly than application

Awareness in Organizations

Knowledge Depot [KRZ97]

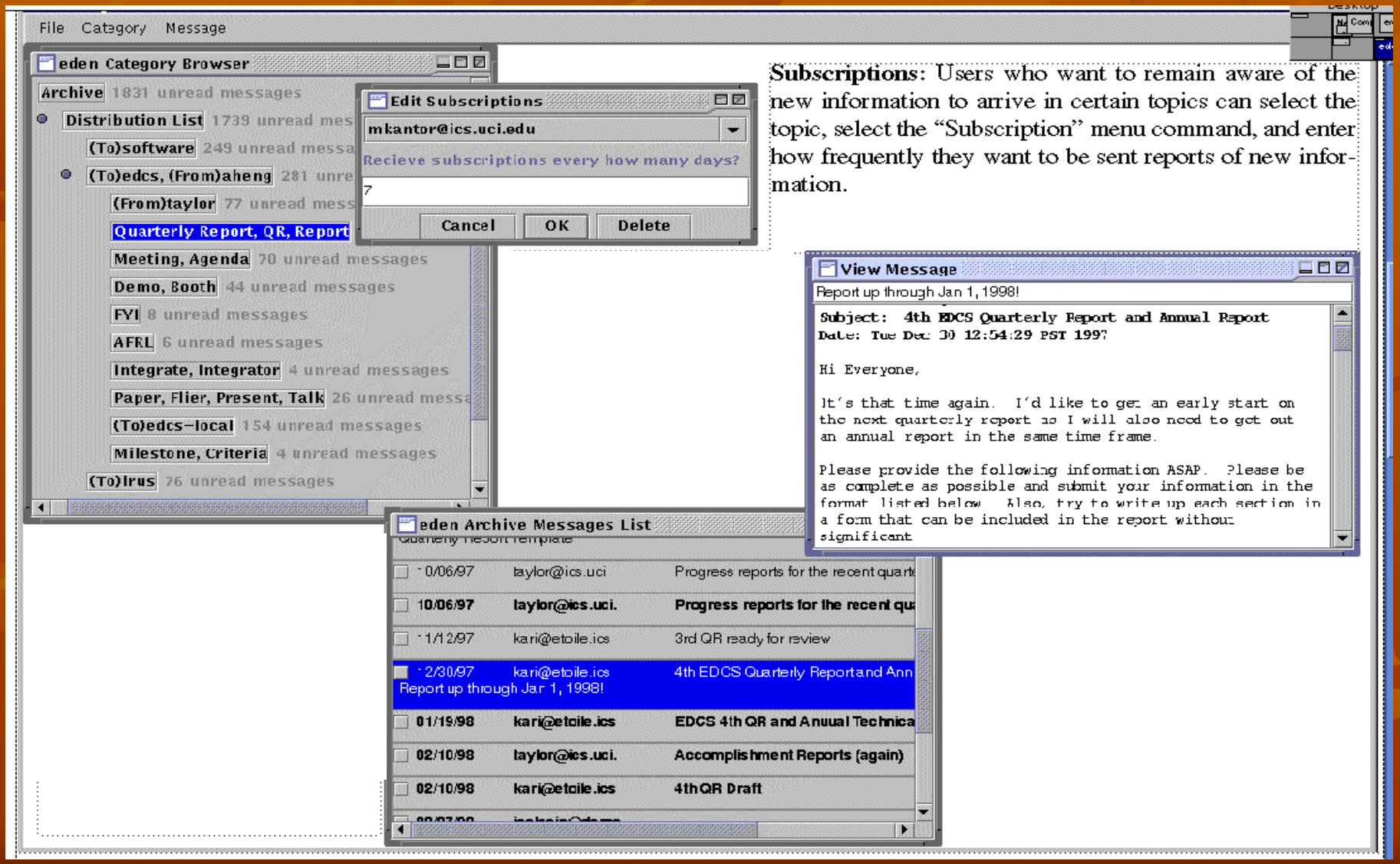
CASSIUS [KR01]

Knowledge Depot [KRZ97]

- Problem: In large organizations, users want to be informed about different topics
- information overload
  - Many e-mails
  - Many mailing lists
  - That have no structure

- Approach: Knowledge depot
  - Categorization: Organize messages according to topics
    - A message can be in more than one topic
    - A topic is a filter in the message content
  - Summarization: Send topic summaries periodically
  - Visual programming (subscription)

Knowledge Depot Interface



CASSIUS Awareness Server [KR01]

Collecting and combining
information from different sources

Information Sources

Physical Environments
(Portholes, Active Badges)

Virtual Environments
(MUDs, chat rooms)

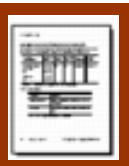
Shared Artifacts
(Papers, spreadsheets, databases)

Mobile Workers
(Customer Rep, Support Staff)

CASSIUS Awareness Server

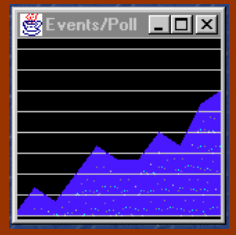
Awareness Tools

Complex tools: Portholes

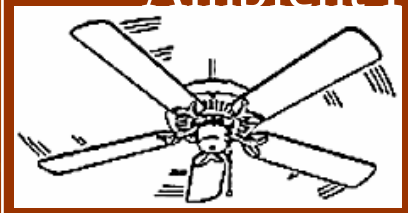


Writer Developer Document Printer

Simple desktop widgets



Ambient Fixtures [ishii]



Mobile Awareness

Harnessing Events

Two Tradeoffs in Subscription

- Subscription Detail-Variety Tradeoff
  - Many details but low variety (e.g. active badges)
  - Few details but high variety (e.g. MUDs)
  - What can I subscribe to and how to manage my subscriptions?
- Lack of Configurability
  - Delivery of information fixed (e.g. by application programmer)
  - However context frequently changes
  - How can I change my application subscription to include/exclude this new/old information?

CASSIUS provides

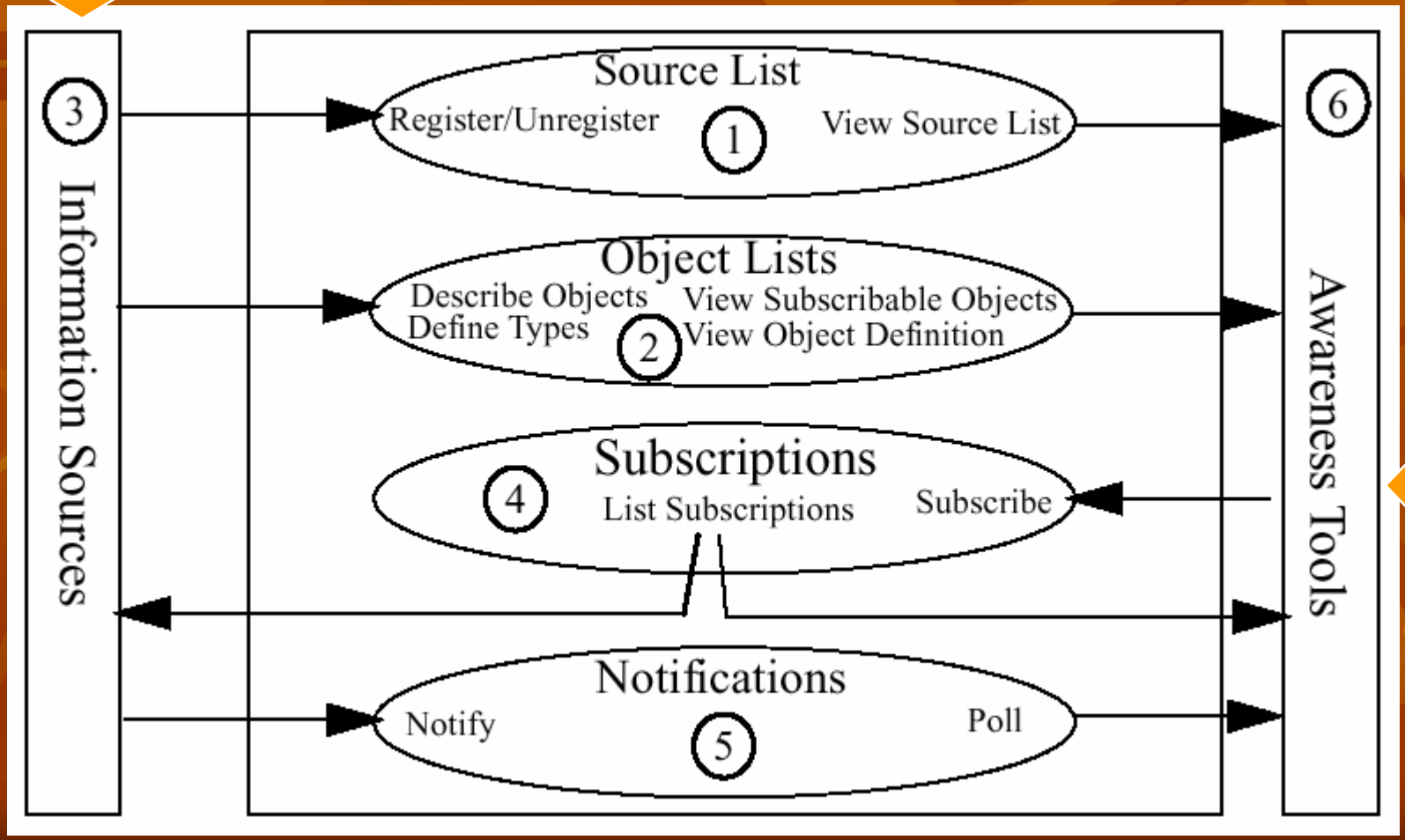
- Awareness information router and integrator.
  - Events are collected from many sources using HTTP protocol
- Ability to define event hierarchies of event sources
- End-user runtime change of subscriptions
  - GUI that supports event hierarchy browsing and subscription edition
- Support for mobility and subscription change:
  - Persistent notifications
    - Can be retrieved later (pull)
  - Persistent subscriptions
    - can also be modified (updated)

CASSIUS [KR01]

Interchangeability and the Detail-Variety Tradeoff

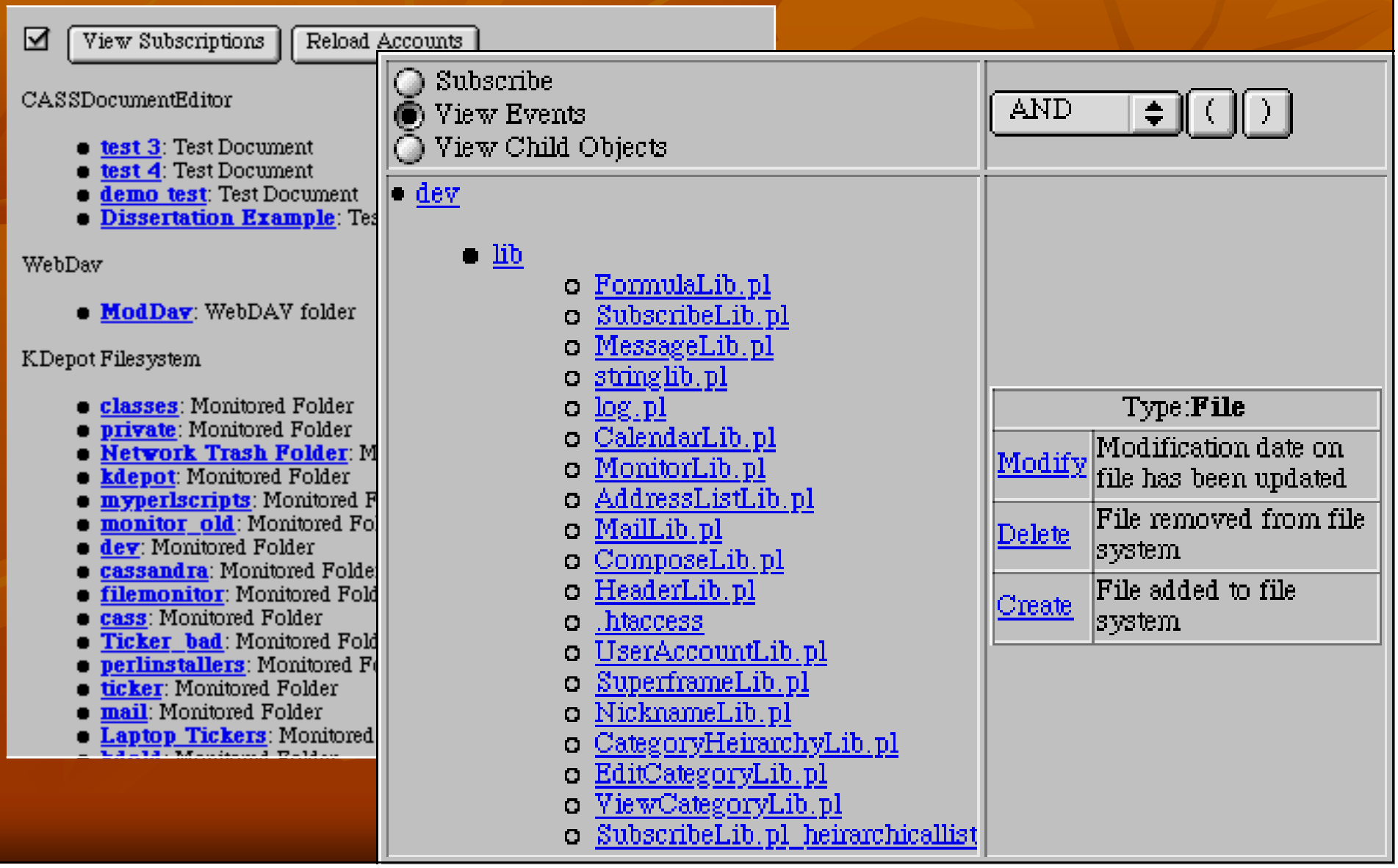
AWACS Simulator

CASSIUS server



Gauges

Managing Subscriptions



Impromptu: security awareness in collaboration

See [DePaula et al. 05]

Security in Collaboration - Problem

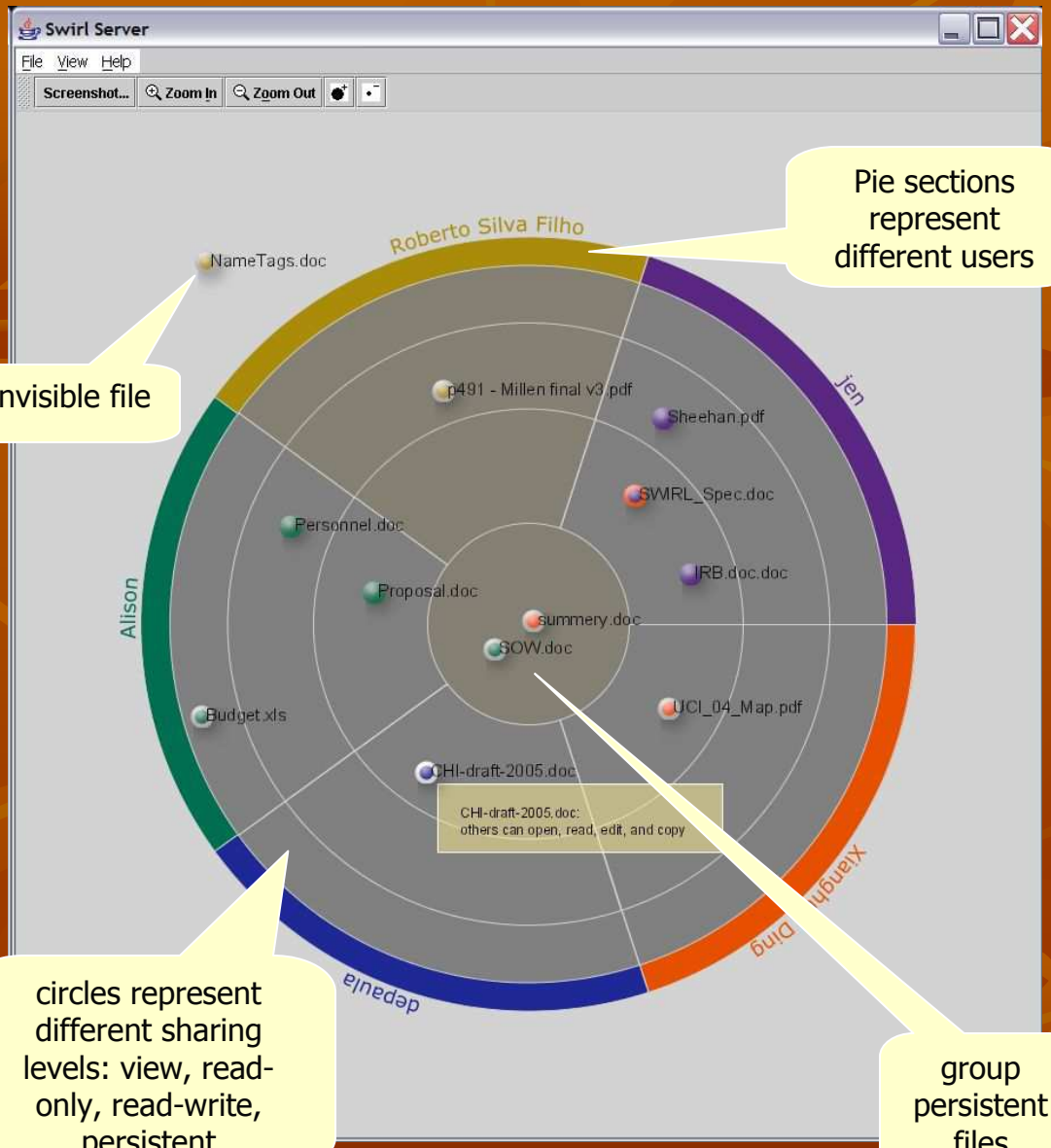
- Trade-offs
  - Secure ←→ useful and trustable
  - Theoretically secure ←→ effectively secure

- As security and privacy mechanisms fade into the background, the harder it becomes for people to understand the result of their actions and whether they will cause privacy and security problems

Approach

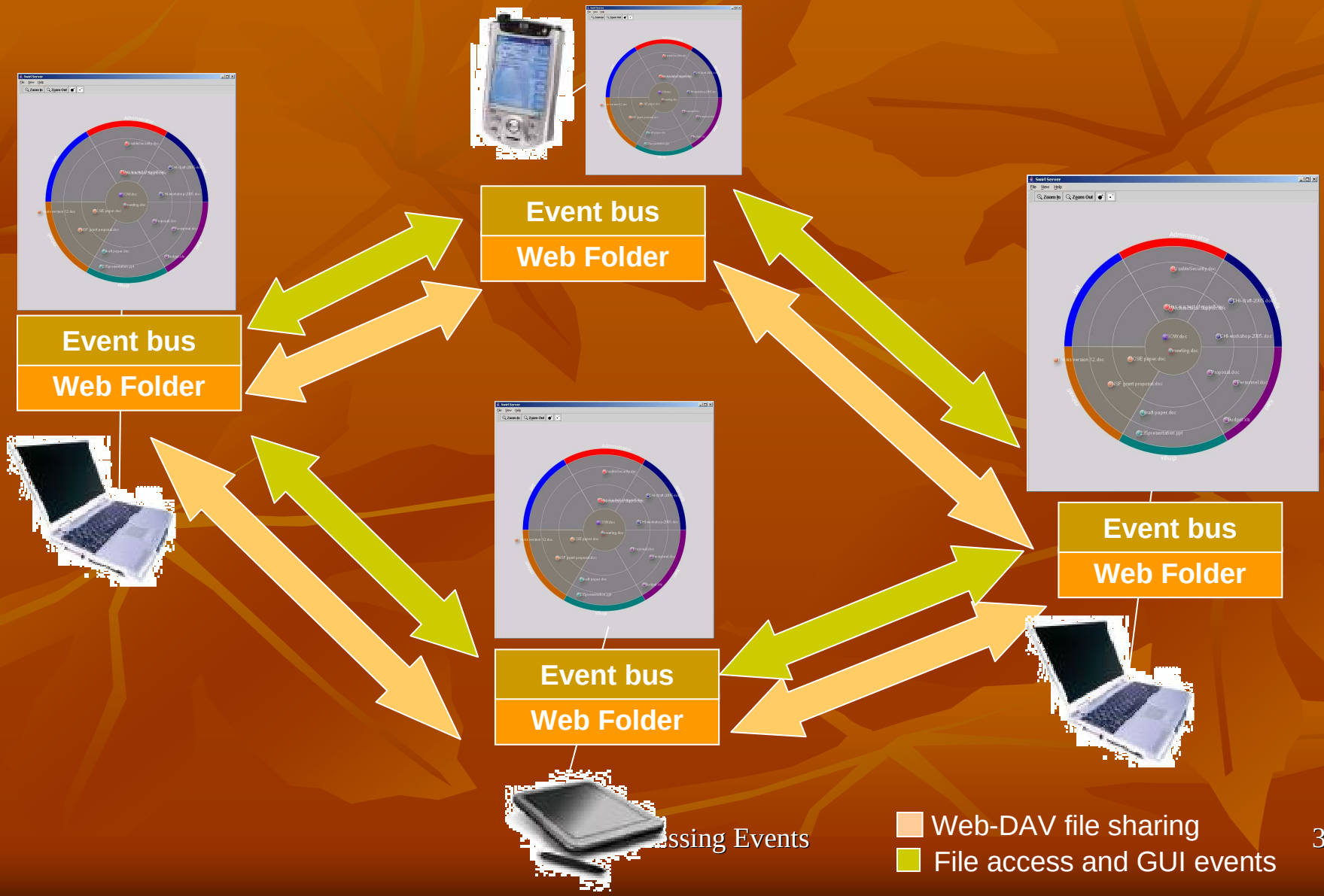
- Making certain privacy and security mechanisms in networked environment visible so as to reveal the consequences of people's actions
- Impromptu
  - A test bed application to assess the combination of activity and configuration with awareness in an ad-hoc collaboration environment

Test bed: Impromptu - interface

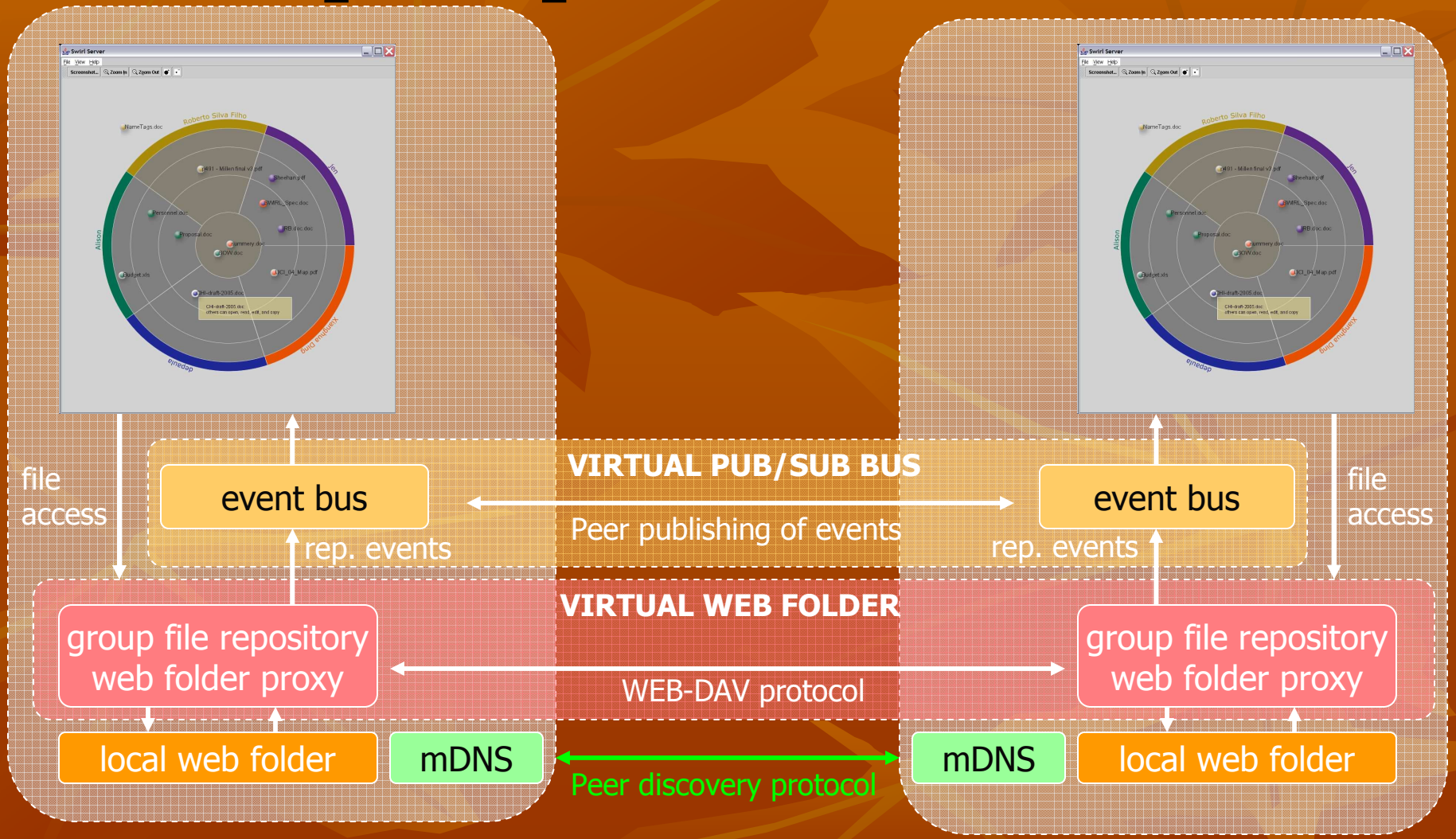


- Different pie slices represent peers in the network
- Users can drag and drop files to be shared
- Concentric circles define different sharing levels
- Files placed in the center are persistent and available to all the members of the group
- Files outside the pie are not shared and become invisible to other peers
- Files blink with the peer color whenever someone reads or modifies it

Impromptu P2P file sharing [DePaula et al. 2005]



Impromptu - architecture

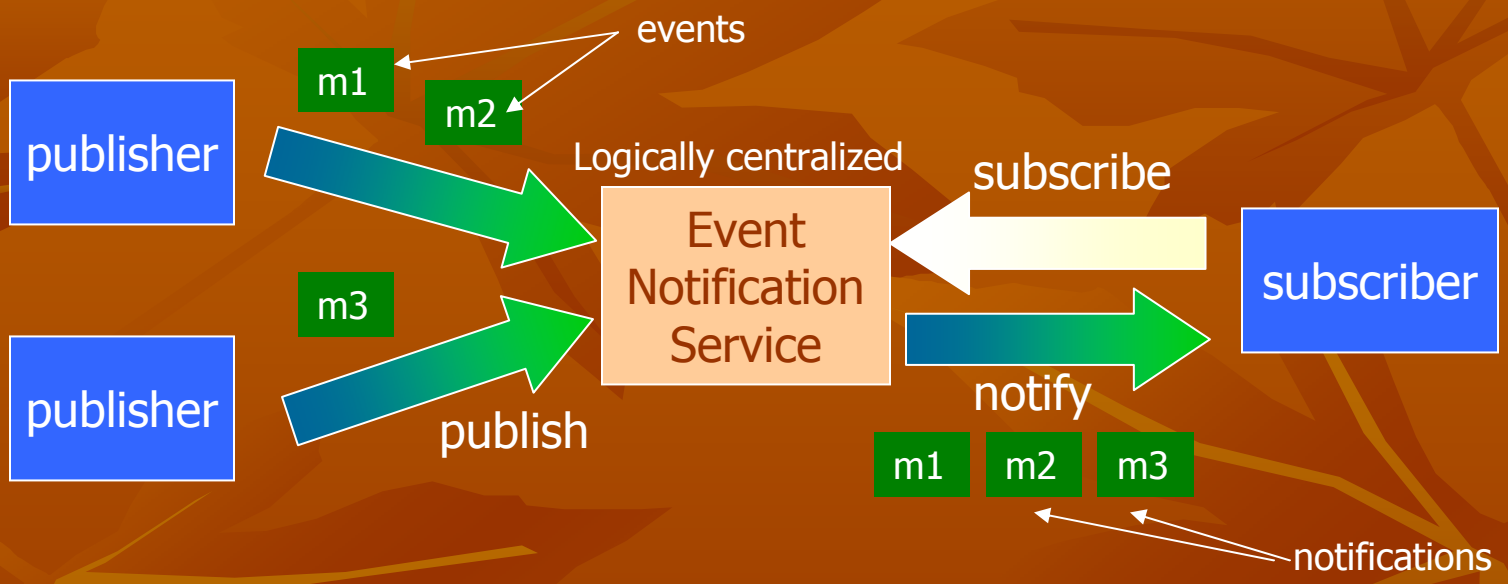


Event Notification Services

Event Notification Servers

- Distributed implementation of the Publish/Subscribe pattern
- Better decoupling of publishers and subscribers
  - One-to-many and many-to-many communication
  - Better anonymity of publishers and subscribers
  - Can provide event filtering and additional services to support different applications
  - Examples:
    - Elvin [FM99]
    - Siena [CRW01]
    - Many others…

E-N Services General Model

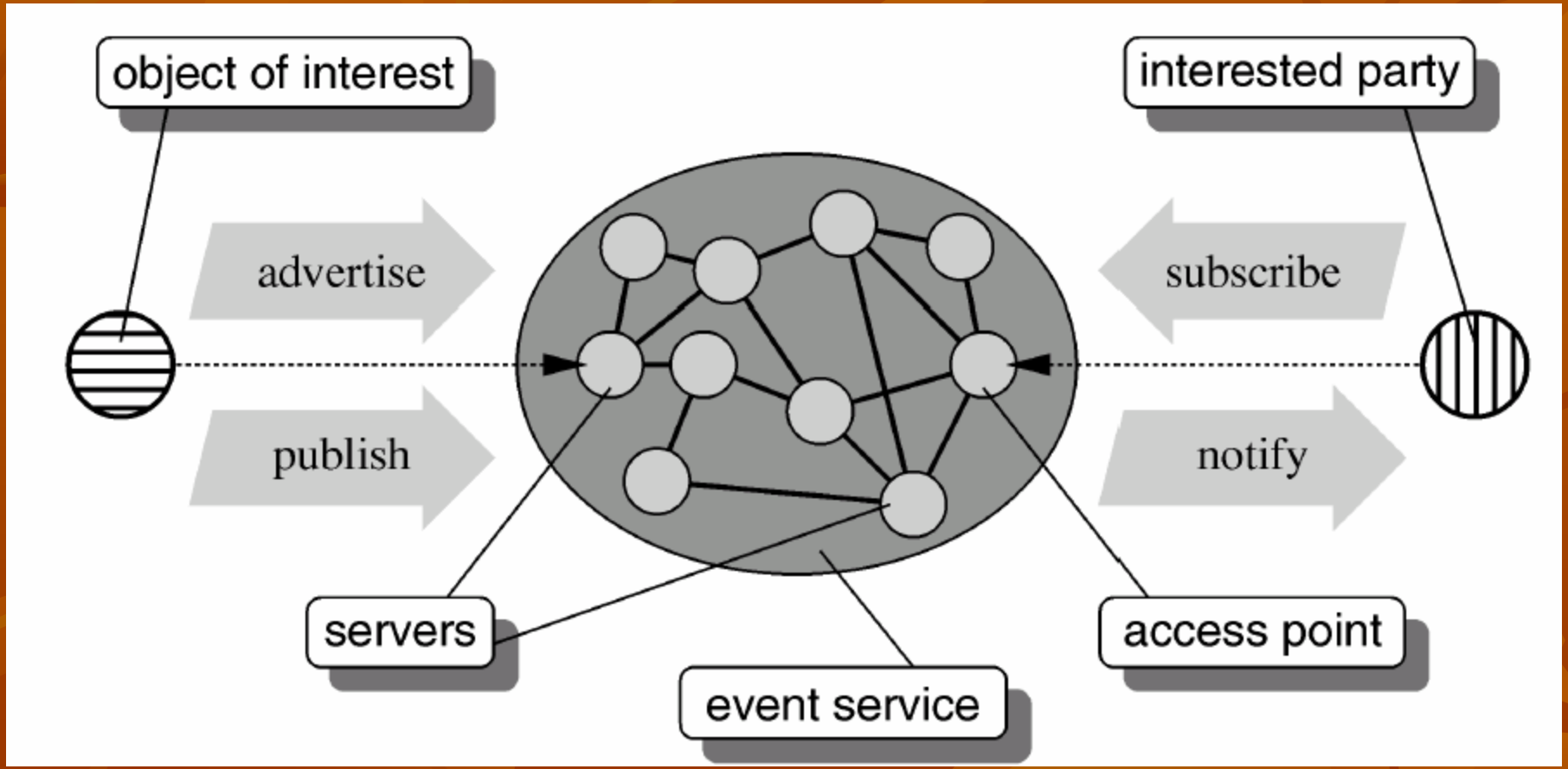


- Publishers generate events (messages)
- Subscribers
  - announce interest on events using subscriptions
  - receive notifications when events match the subscription

Siena [CRW01]
Content-based network

- Challenges
  - Internet-scale content-based routing network
  - Trade-off: expressiveness versus scalability
- Solution
  - Topology: general graph of routers
  - Algorithm: subscription forwarding (advertisement) and merging (based on coverage)
  - Routing: based on shortest path (hops)

Siena overview



Siena main features

- A network of content-based routers that provide
  - Internet-scale distribution
  - Content-based filtering (subscription) capability
    - Events are represented as (attribute/value) pairs.
    - Filter capability
      - Content operators as > < == <= and >=
      - String matching: ‘*’
      - Basic types: string, int, long, boolean, double.
  - Those benefits result in some open issues…

Internet-scale content-based routing: open problems [Rosenblum2003,2005]

- There is no killer application to Siena (yet)
  - Most applications are localized (clustered)
  - Who really needs internet-scale events?
- How to add security to content-based routing?
  - How to route events based on their content without reading its content?
  - How to prevent denial of service attacks? Subscribe to '*' (all events), for example.
- Mobility of publishers and subscribers
  - How to recalculate the routing?
  - Where to store the notifications while subscribers are off-line?

Generalized infrastructures

Can notification servers as Siena support all applications?

- The answer is NO!
- Siena focus on scalability and routing issues

- Many of the awareness, monitoring and collaboration features are not supported:
  - mobility and event source hierarchy browsing, pull notification, end-user runtime subscription change (CASSIUS)
  - Advanced event processing as aggregation and abstraction (EDEM)
  - Event summarization capability (EDEM, Knowledge Depot)
  - Peer-to-peer support (Impromptu)
  - And many others…

YANCEES [SDR03]

Research questions:

- Can we build a flexible (extensible and configurable) infrastructure to support all those applications?
- How flexible should it be?
- What are the trade-offs involved?

- In other words, can we build a flexible pub/sub service that supports different application requirements?

- Our prototype: YANCEES
  - Yet Another Configurable and Extensible Event Service

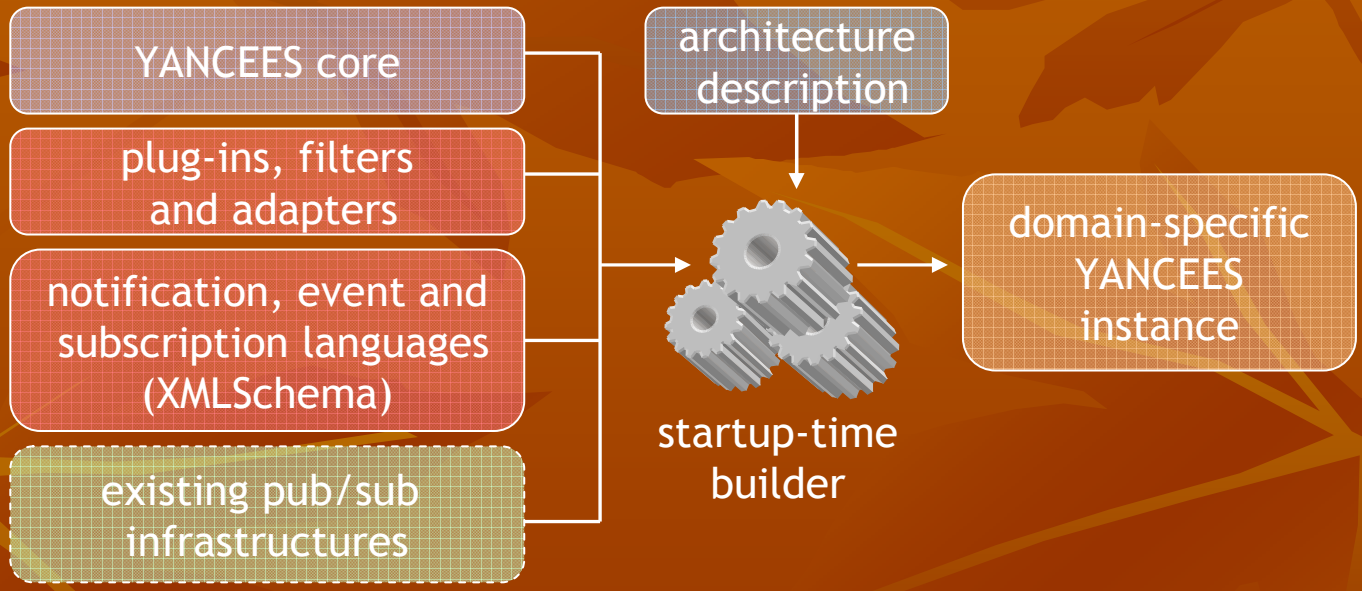
YANCEES current approach

- Separate a pub/sub infrastructure into its basic design dimensions:
  - Notification, Subscription, Routing, Event representation and Protocols
- Combine:
  - Extensible languages (XML) and plug-ins
  - Filters
  - Adapters
- Thorough a configuration manager
- Over a common framework

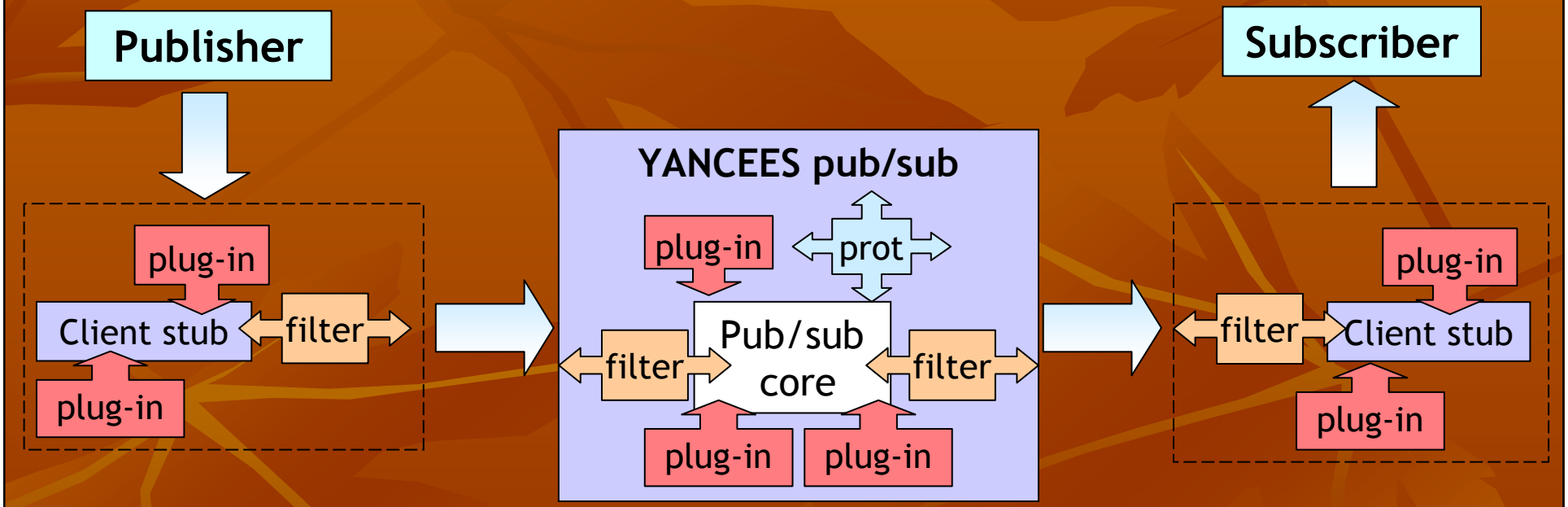
Publish/subscribe design dimensions

Dimension	Definition	Example
Subscription	specifies how subscribers express interest in subsets of events	content-based, topic-based, advanced event processing
Notification	specifies how notifications are delivered to subscribers	push, pull, both, others.
Event	Specifies how events are represented	tuple-based, record-based, XML documents, e-mail text
Protocol	other kinds of interaction with the service	**Interaction protocols**: authentication, manual roaming **Infrastructure protocols**: federation, replication, fault-tolerance
Resource	defines where in the system (publishers/subscribers/routers) the extensions are placed	client-side (as EDEM), server-side (as CASSIUS)

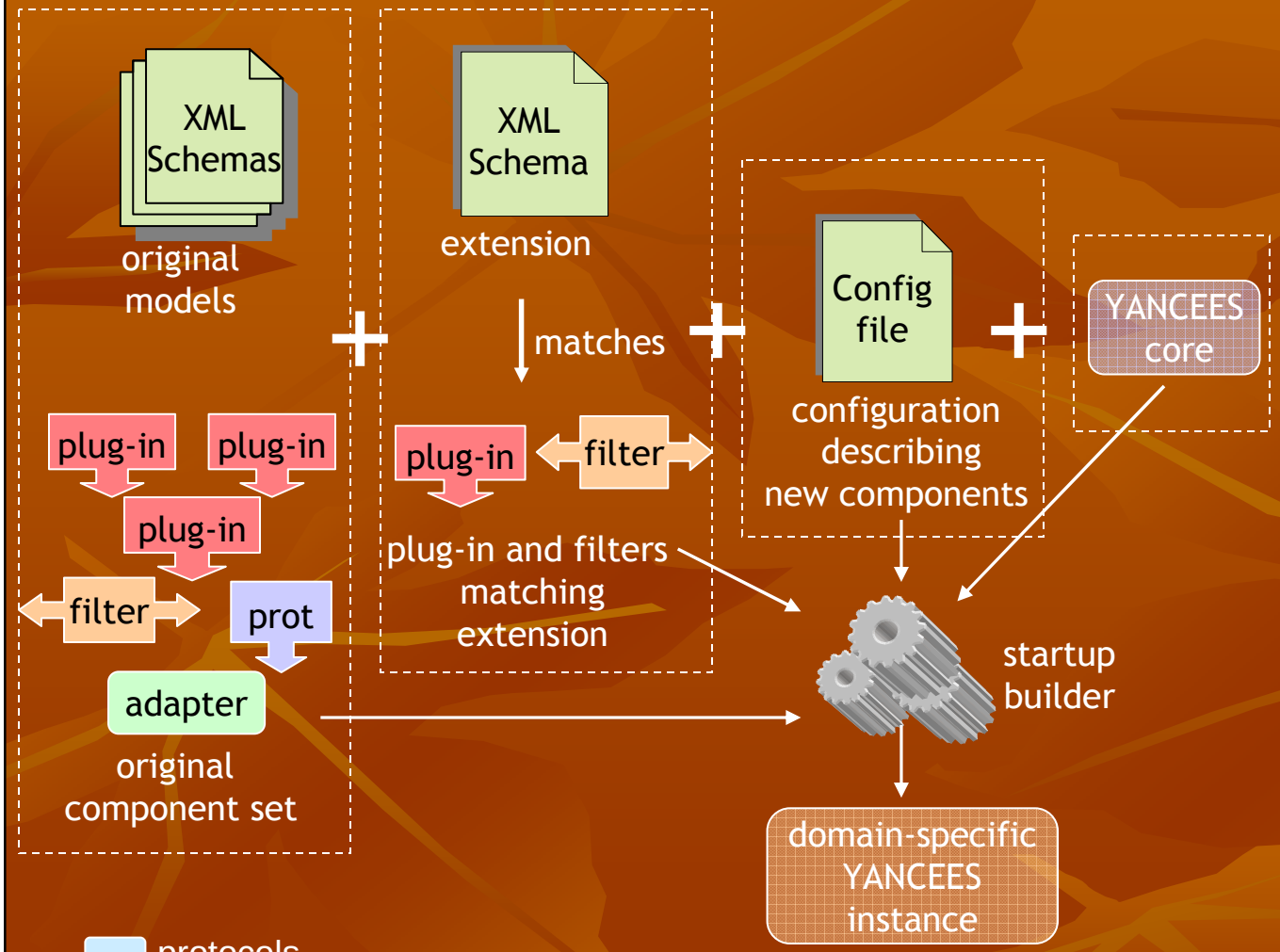
Approach summary



YANCEES Framework



How to extend YANCEES?



1. Define a language extension using XML Schema
2. Implement plug-ins and filters for that extension
3. register plug-in in the configuration file
4. Restart the server with the new configuration

Positive Experiences (so far)

- Different applications were supported
  - Event monitoring (Swirl)
  - Peer-to-peer collaboration (Impromptu)
- The ability to add input and output filters and to change the event matching algorithms and protocol plug-ins make it easy to customize the infrastructure
- However, there are costs…

Current Issues

- Framework costs:
  - Initial generalization and implementation
  - Initial learning curve (not much worse than more advanced pub/sub systems as CORBA-NS)
- Performance:
  - The XML extensibility and the framework add extra overhead (around 100 ms)
- Interdependencies
  - Hard to co-evolve the subscription language with the infrastructure
  - Crosscutting concerns
    - Event model imposes a data coupling in all dimensions
- Usability:
  - XML is not a good subscription language
  - Depending on the feature to be added, too many points need to be adapted/extended

Future work

- Re-think (design) the architecture
  - Restrict the variability dimensions
  - Modularize the main concerns
  - Automate language extension and programming
- Apply separation of concerns techniques:
  - AOP
  - Agent technology
  - Automatic subscription generators
  - others…
- Improve overall usability

Conclusions

- Event-based interaction provide an interesting paradigm for software engineering
  - Decoupling
  - Integration
  - Anonymity
  - Multicast communication
- Different applications can benefit from this paradigm. Some examples: EDEM, Awareness, Groupware and Software Engineering
- Many issues are still open:
  - Event-based infrastructures (Siena, YANCEES)
  - Event-based interaction (usability, integration, application support)

Questions / comments

References

References

- [JL01] H. A. Jacobsen and F. Llibart. Tutorial on Publish/subscribe Systems. Tutorial delivered at the 17th International Conference on Data Engineering, April 2001, Heidelberg, Germany. Presentation:http://www.eecg.toronto.edu/~jacobsen/courses/icde01/sld001.htm
- [GHJV95] Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Reading Mass., Addison Wesley.
- [CRW01] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf, Design and Evaluation of a Wide-Area Event Notification Service, ACM Transactions on Computer Systems, vol. 9, no. 3, August 2001, pp. 332-383.
- [HR98] Hilbert, D., Redmiles, D. An Approach to Large-Scale Collection of Application Usage Data Over the Internet, Proceedings of the Twentieth International Conference on Software Engineering (ICSE '98, Kyoto, Japan), IEEE Computer Society Press, April 19-25, 1998, pp. 136-145.
- [TMA+96] R. Taylor, N. Medvidovic, K. Anderson, E.J. Whitehead, J. Robbins. "A Component- and Message-Based Architectural Style for GUI Software," IEEE Transactions on Software Engineering, June 1996.
- [SNH03] Anita Sarma , Zahra Noroozi and André van der Hoek , Palantír: Raising Awareness among Configuration Management Workspaces . In Proceedings of Twenty-Fifth International Conference on Software Engineering,pp 444-454, May 2003, Portland, Oregon.
- [KR01] Kantor, M., Redmiles, D. Creating an Infrastructure for Ubiquitous Awareness, Eight IFIP TC 13 Conference on Human-Computer Interaction (INTERACT 2001-Tokyo, Japan), July 2001, pp. 431-438.

References (cont)

- [DePaula+05] DePaula, R., Ding, X., Dourish, P., Nies, K., Pillet, B., Redmiles, D., Ren, J., Rode, J., and Silva Filho, R. S. In the Eye of the Beholder: A Visualization-based Approach to Information System Security. International Journal of Human-Computer Studies - Special Issue on HCI Research in Privacy and Security, Vol. 63 , Issue 1-2, pp. 5-24. July 2005.
- [SDR03] Silva Filho R. S., De Souza C. R. B., Redmiles D. F. The Design of a Configurable, Extensible and Dynamic Notification Service. in Proc. Second International Workshop on Distributed Event-Based Systems (DEBS'03), In conjunction with The ACM SIGMOD/PODS Conference, San Diego, CA, USA, pp.1-8, June 8th, 2003.
- [CN01] Cugola, G., E. D. Nitto, et al. (2001). "The Jedi Event-Based Infrastructure and Its Application on the De-velopment of the OPSS WFMS." IEEE Transactions on Software Engineering 27(9): 827-849.
- [Rosenblum2003] Rosenblum, D. Some Open Problems in Publish/Subscribe Networking (keynote presentation at DEBS'03)
- [Rosenblum2005] Rosenblum, D. Content-Based Publish/Subscribe: A Re-Assessment (keynote presentation at DOA 2005)
- [FM99] Fitzpatrick, G., T. Mansfield, et al. 1999. Augmenting the workaday world with Elvin, Proceedings of 6th European Conference on Computer Supported Cooperative Work ECSCW'99, 431-450.
- [GLT99] Girgensohn, A., Lee, A., Turner, T. Being in Public and Reciprocity: Design for Portholes and User Preference, In Human-Computer Interaction INTERACT '99, IOS Press, pp. 458-465, 1999.