ICS 221 – Harnessing Events

Overview

- Motivating Applications
  - Software Engineering
  - Awareness in Organizations and Ubiquitous Delivery
- Current Research
  - Striving for Versatility
- Vision of the Future
  - Striving for Unified Infrastructure
- Related Work

Events and Software Engineering – One Scenario

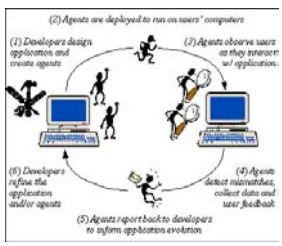
See [HR98]

Background

- 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.
- Our research: techniques to enable large-scale incorporation of usage data and user feedback in 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

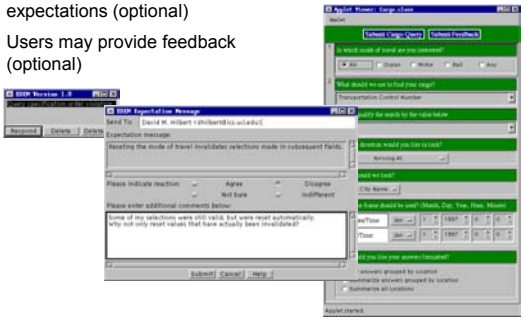
- A cargo query form

Agents

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

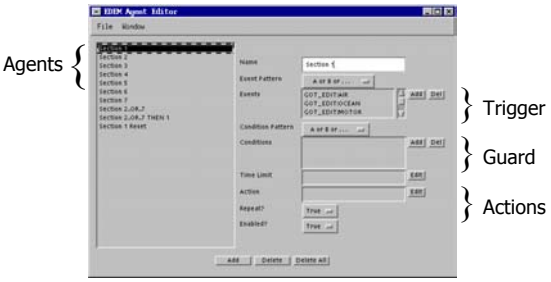
Users

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



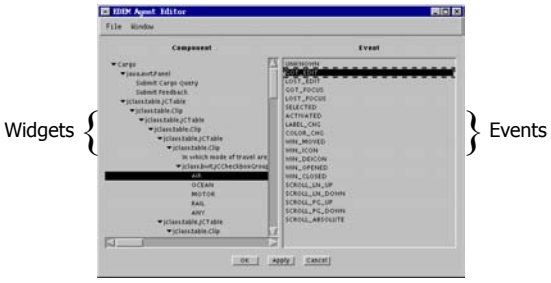
Agent Authoring

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

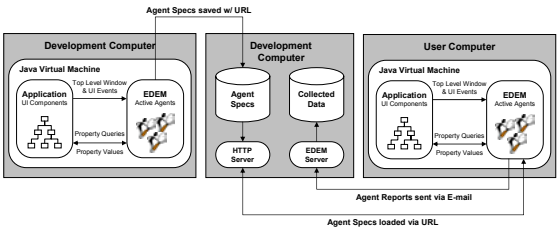


Event Specification

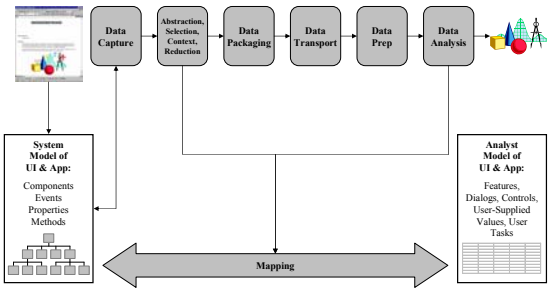
- Detecting when the user selects “AIR” as the “mode of travel”



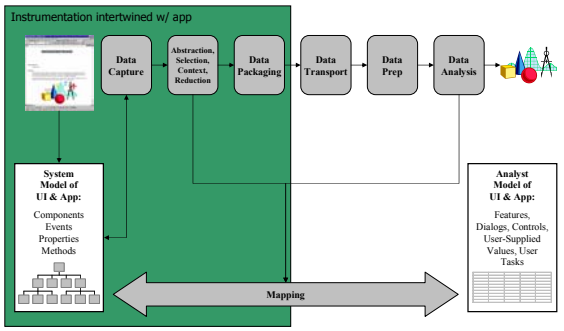
Architecture



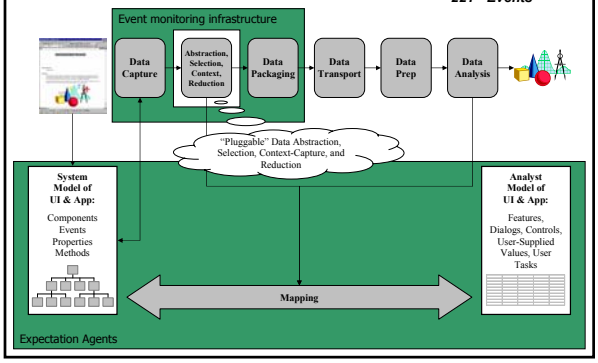
Reference Architecture (see ACM Computing Surveys)



Reference Architecture Traditional Instrumentation



Reference Architecture (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

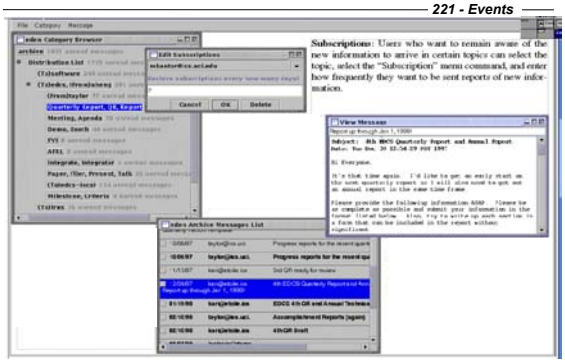
Other Possible Applications

- 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 in which
  - event and state information can be easily "tapped"
  - low-level data must be related to higher level concepts of interest
  - available information exceeds that which can practically be collected
  - data collection needs evolve over time more quickly than application

Awareness in Organizations

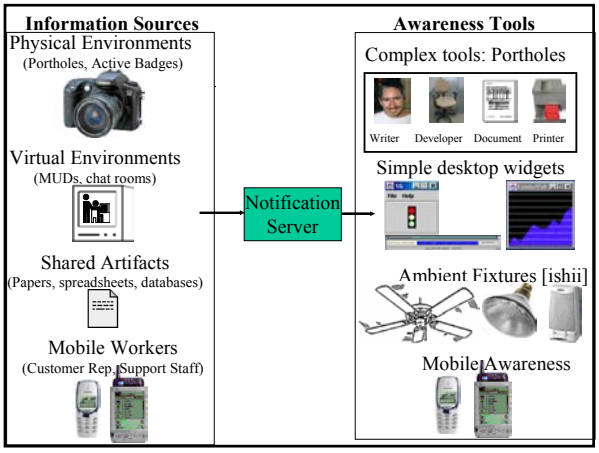
See [KRZ97], [KR01], and [deS+02]

Knowledge Depot [KRZ97]



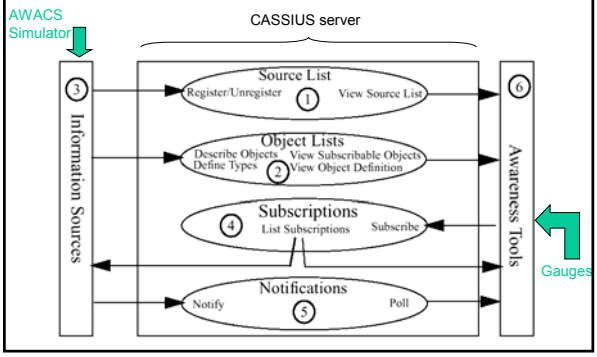
Two Tradeoffs in Subscription

- Detail-Variety Tradeoff
  - Many details but low variety (e.g. active badge)
  - Few details but high variety (e.g. MUDs)
- Lack of Interchangeability
  - Delivery of information fixed (e.g. by applications programmer)
  - However context frequently changes

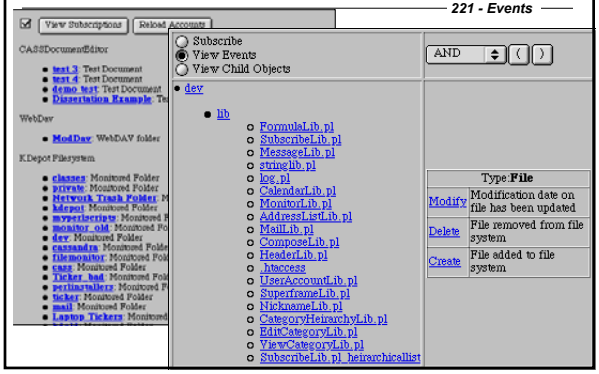


CASSIUS [KR01]

Interchangeability and the Detail-Variety Tradeoff



Managing Subscriptions



Issues [deS+02]

1. How gauges are notified about the events or The Issue of Push vs. Pull Architectures between the notification server and the gauges?
2. How powerful is the subscription service for each notification server? What are the types of matching supported?
3. Which objects can send events to the notification server, or Issues about event and object registration?
4. Which meta-information is associated to the events sent to the notification server or How powerful are the events?
5. What are the interfaces implemented by the notification servers, or how easy to change from a notification server to another?

What More is Needed?

- Meta-information about events and objects
  - Including classification hierarchy and synonyms
- Account (producer/consumer) registration
- Ideally, variable distribution of services from producer to consumer
- Ideally, heterogeneous federations of servers
- Usability testing on APIs and designers
- Usability testing on designed consumer applications

…

Notification – Push vs. Pull

- Whether consumers are sent events directly or they poll for events.
- Depending on the model, a layer or server in between the notification server and the consumers (e.g., gauges) needs to be implemented
  - E.g., to adapt CASSIUS to have push qualities
  - E.g., to adapt Elvin and Siena to have pull (e.g., caching) qualities

Notification – Push vs. Pull – Lessons

- Knowing exactly what model is in use in not always obvious.
  - Mixed models are often in use
- The mismatch between architectural models and application requirements can lead to
  - additional design and programming effort
  - design and programming errors
- Future notification servers may combine service models to simplify application design.

Power of the Subscription Service

- How consumers can select events of interest
  - Logical expressions, regular expressions, sequences, and compound expressions
- and where the selection takes place
  - Information source
    - The consumer specifies the subscription, which the notification servers passes to the information source, which performs the selection.
  - Notification server
    - The consumer specifies the subscription and the notification server performs the selection.
  - Consumer
    - The consumer receives all events and performs the selection.

Subscriptions – How and Where

	Source	Server	Consumer
Logical		CASSIUS Elvin Siena	
Regular or (Substring)		Elvin (Siena)	CASSIUS
Sequence		Siena	CASSIUS Elvin
Compound			CASSIUS Elvin Siena

Subscription Service – Lessons

- For supporting awareness, the power of the subscription service determines the power of the awareness device.
  - I.e., whether the server supports sophisticated subscription specifications or if sophistication needs to be added at the information sources or consumers.
  - (also increasing inflexibility of design and likelihood of errors)
- Depending on the distribution of the service …
  - Traffic is decreased the closer to the source the subscription service is performed.
  - However, coupling is increased if the information source is involved in performing the subscription service (whether hard-coded or specified)

Event and Object Registration

- How much information an event notification server represents about information sources and events.
- Whether or not information sources need to inform the notification server of their object type, object hierarchy, and event types.
  - CASSIUS uses event and object registration, and maintains a hierarchy of objects.
  - Elvin and Siena do not. ☺

Event and Object Registration– Lessons

- If a notification server supports event and object registration, and maintains a hierarchy of objects, then …
  - Consumers (gauges) are more decoupled from the information sources (I.e., they can browse the notification server for the names of objects and events and not be hardwired to specific information sources – reminiscent of MVC separation of concerns)
- Notification servers that do not support registration …
  - Do not allow consumers (gauges) to browse potential object hierarchies and event types
    - Limiting the power of the consumers as well
    - Increasing the coupling between information sources and consumers
  - May miss events due to name mismatches, e.g.,
    - Different information sources may simply use different names for what is actually the same event, or
    - Different names may have reflected sub-typing

Event Meta-Information

- Additional information about an event that is sent to the consumers (e.g., timestamps, information source, event type)
- Depending on the server's design, meta-information could be added by either the server or the information source
  - Using CASSIUS
    - The event type may be included by the information source
    - The server checks the event type (based on the registered object hierarchy) and adds a timestamp
    - (The information source may also include its separate timestamp.)
  - Using Elvin and Siena
    - Meta information must be provided by information sources without the server augmenting the event or performing checks.
    - Meta information is mixed in with the event information.

Event Meta-Information – Lessons

- Systems built with servers that support meta-information
  - may have fewer errors because of
    - Consistency checks based on event and object types
    - Not mixing of meta- and content information in events
  - and allow more powerful awareness devices because of the added meta information.
- Servers that support meta-information may be extended more readily to provide additional services such as guaranteed delivery, security, and others
  - without tight coupling between sources and consumers.

Interfaces, APIs, Services

- How event data is moved to consumers.
- Depending on the model, a specific interface (method) is used:
  - Designers of a pull notification server specify a method for polling (and other services)
  - Designers of a push notification server specify a method for consumers to implement to receive events (and other services)
  - (of course specific interface names and parameters vary from server to server)

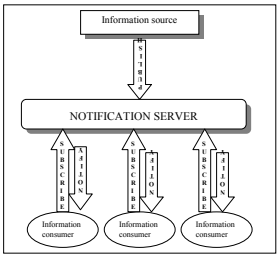
Interfaces, APIs, Services – Lessons

- Ease of change
  - Consumers (gauges) must be modified (not surprisingly) to adapt to different servers
- Heterogeneous architectures
  - When multiple notification servers are in use an N-N problem exists between gauges and servers (as well as between information sources and servers)
  - One solution is to design systems with consumers linked always to the same server and connectivity achieved through server interconnections.
- Designers of gauges must apply additional effort to design for modification and heterogeneity (in lieu of a standard)

Current Research and Visions of the Future

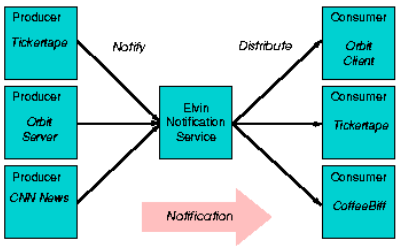
Event Notification Service

- Information Sources
- Information Consumers (Gauges)
- Event Notification Servers
- Event Services
  - Publish (Post)
  - Notify (Receive)
  - Subscribe



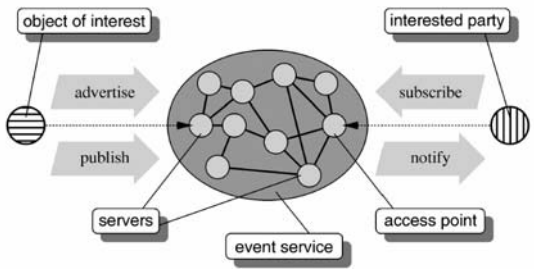
Elvin [FM+99]

Basic Notification Service

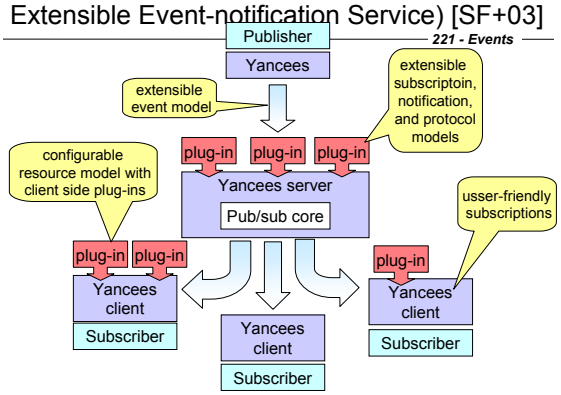


Siena [CRW01]

Scalability in a Global Distributed Environment



Yancees (Yet Another Configurable and Extensible Event-notification Service) [SF+03]



New ideas

- Addresses the extensibility problem in publish/subscribe services using
  - Extensible models and languages (XML)
  - Plug-ins
  - Dynamic message parsers
- Fine-grained extensibility based on event, notification, subscription and protocol models
- Builds on top of current publish/subscribe networks (Elvin, Siena, CORBA and so on)

Impact

- Better collaborative and distributed application support
  - Moves the burden of event processing from the application to the middleware
  - Faster extension of server functionality
  - Load balancing: client-side and server-side plug-ins
- Runtime change and upgrade of plug-ins
- Improved integration of applications
- Built on top of different publish/subscribe networks (Elvin, Siena and others)
- Improved usability with client-side plug-ins and GUI subscription controls

Schedule

- Design (06/02-03/03)
- Implementation of core functionality (04/03-12/03)
- Integration with current applications and software development tools (Eclipse) (01/04 – 06/04)
- Additional plug-in implementation (01/04 - 06/04)
- Development of collaborative applications with end users feedback (06/04-09/04)

Future Vision

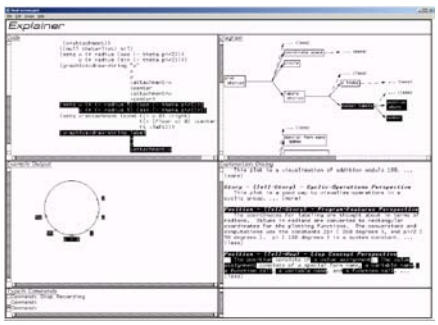
Example: Group Awareness through a Portholes System[GLT99]

- Shows presence of collaborators and relevant spaces
- Uses visual cues (such as this theater style) to condense view according to relevance.

Girgensohn / Lee / Turner 99

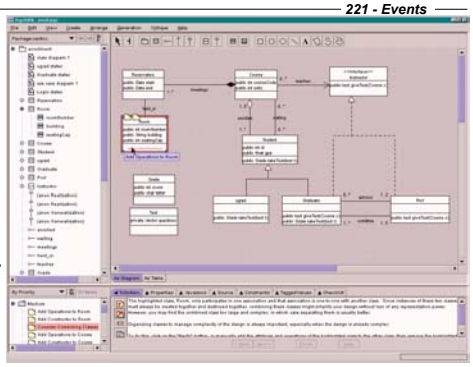
Explainer [Red93]

- Early Hypermedia for human learning by example
- Extreme hyper-granularity
- Incremental [Minimal] Explanation
- Human Variability Reduced



Argo/UML [RHR98] [RR00]

While designers work, design critics analyze the design and provide helpful advice. The "to do" list (lower left) presents and organizes advice about pending design changes.



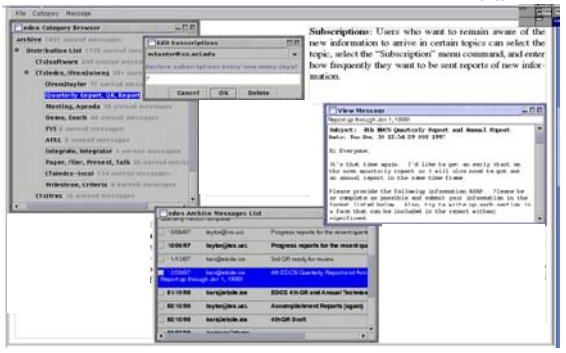
EDEM – Expectation-Driven Event Monitoring [HR98]

Agents monitor application events

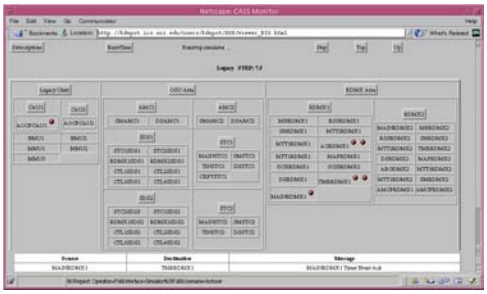
And optionally facilitate feedback.

Knowledge Depot [KRZ97]

Kantor, Redmiles, Zimmermann 97



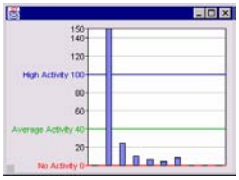
Gauges for Application Awareness [deS+02]



Architectural Message Passing Monitor

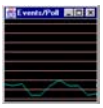
Some More Gauges

a) Progress Bar



b) Bar Chart

f) Signal



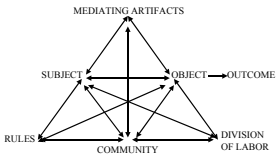
c) Line Graph

d) Clock

e) Load

Activity Theory and Design [Red02] [CSR02]

- Identify the stakeholders in the process.
- Help ensure that technology is designed to the users, other stakeholders, and the organization.
- Work toward alignment between users' rewards and business' needs.
- Work toward alignment between the rewards of the designers of the device and both the end users' and business' needs.



Engeström [Eng90] Activity System Model

Theme - Awareness Information

- Argo/UML
  - Critics notify end users of design problems
- EDEM
  - Agents monitor application usage and report data to designers
- Knowledge Depot
  - End users subscribe to email categories / topics
- Gauges
  - "Probes" (instrumentation) should collect specific information about distributed applications' behavior and performance and supply this information to narrow-purposed "Gauges" (visualizations)
- Activity Theory
  - Many people and things affect the achievement of an objective.

Conclusions

Conclusions

- The available software (e.g., notification servers) for building systems incorporating awareness information is very low-level and prone to design and programming errors.
- Support for complex, heterogeneous systems (e.g., multiple different, servers, information sources, and consumers) varies, currently designers must expend extra effort to design for change and flexibility.
- Although programmers acclaimed our selection of gauges, a grounded selection of gauges eventually will require empirical evaluation.

Integration

- A greater variety of awareness *devices …*
  - Critics
  - Usability expectations
  - Email notifications
  - Application gauges
  - *Security and privacy gauges?*
  - *Portholes?*

… integrated through an event notification infrastructure

Final Point

- The goal is usability—we seek to provide a set of usable and useful services and strategy to provide usable and useful awareness capabilities for applications.

References

References (1 of 2)

- Dourish, P. and Bellotti, V. Awareness and Coordination in Shared Workspaces. Proc. ACM Conf. Computer-Supported Cooperative Work CSCW'92, New York: ACM, 1992, pp. 107-114.
- Carzaniga, A., Rosenblum, D., and Wolf, A. 2001. Design and Evaluation of a Wide-Area Notification Service. ACM Trans. Computer Systems, 19(3), 332-383.
- Collins, P., Shukla, S., Redmiles, D. Activity Theory and System Design: A View from the Trenches, Computer-supported Cooperative Work, Special Issue on Activity Theory and the Practice of Design, Vol. 11, No. 1-2, 2002, pp. 55-80.
- de Souza,C.R.B., Basaveswara, S.D., Redmiles, D. Using Event Notification Servers to Support Application Awareness, to appear in the Proceedings of the IASTED International Conference on Software Engineering and Applications (Cambridge, MA), November 2002, to appear.
- 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.
- 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.
- 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.
- Kantor, M., Zimmermann, B., Redmiles, D. From Group Memory to Project Awareness Through Use of the Knowledge Depot, Proceedings of the 1997 California Software Symposium (Irvine, CA), UCI Irvine Research Unit in Software, Irvine, CA, November 7, 1997, pp. 19-26.
- 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 (2 of 2)

- Redmiles, D. Reducing the Variability of Programmers' Performance Through Explained Examples, Human Factors in Computing Systems, INTERCHI '93 Conference Proceedings (Amsterdam, The Netherlands), ACM, April 1993, pp. 67-73.
- Redmiles, D. Introduction to the Special Issue of CSCW on Activity Theory and the Practice of Design, Computer-supported Cooperative Work, Special Issue on Activity Theory and the Practice of Design, Vol. 11, No. 1-2, 2002, pp. 1-11.
- Robbins, J., Hilbert, D., Redmiles, D. Extending Design Environments to Software Architecture Design, Automated Software Engineering, Vol. 5, No. 3, July 1998, pp. 261-290
- Robbins, J., Redmiles, D. Cognitive Support, UML Adherence, and XMI Interchange in Argo/UML, Information and Software Technology, Vol. 42, No.2, January 2000, pp.79-89.
- Silva Filho, R. S., de Souza, C.R.B., Redmiles, D.F. The Design of a Configurable, Extensible and Dynamic Notification Service, Second International Workshop on Distributed Event-Based Systems (DEBS 2003), In conjunction with The ACM SIGMOD/PODS Conference (San Diego, CA), June 2003, available at http://www.eecg.toronto.edu/debs03/papers/silva_filho_etal_debs03.pdf.

Extra Slides

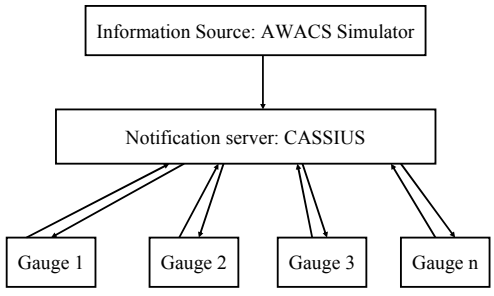
Problem - Monitoring Distributed Information

- Part of the DARPA DASADA project called for the creation of "Probes" (instrumentation) that would collect specific information about distributed applications' behavior and performance and supply this information to narrow-purposed "Gauges" (visualizations)
- In general, the state of a complex distributed system is reflected by event traffic, but the number of events at any given moment make it nearly impossible for people (or other systems) to be aware of critical situations or make decisions for dynamic adaptation.
- A specific example system system in the DASADA project is the AWACS simulator.

Awareness

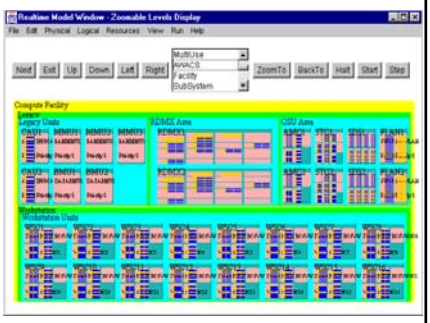
- In general, awareness means having information about other activities that affects a person's own work [DB92].
- Some types of awareness
  - Group awareness
    - Who is around and what roughly are they doing?
    - e.g., images relayed in Portholes
  - Project awareness
    - What knowledge affects (e.g., decisions are made about) project content?
    - e.g., subscriptions in Knowledge Depot
  - Application awareness
    - What's going on in running software?
    - E.g., architectural gauge

Using the CASS Strategy

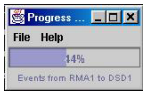


Application - the AWACS Simulator

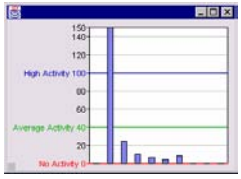
- 130+ components
- 200+ connectors
- 30+ subsystems
- About 15 events per clock increment (we were not told correspondence to real time)



Some Gauges [SBR02]

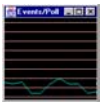


a) Progress Bar



b) Bar Chart

f) Signal

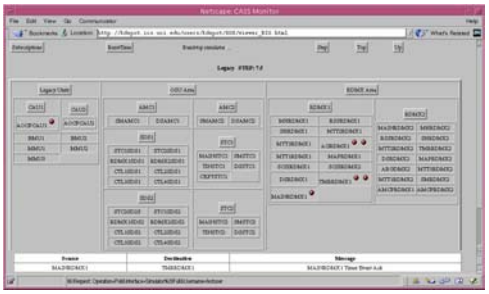


c) Line Graph

d) Clock

e) Load

Another Kind of Gauge



g) Architectural Message Passing Monitor

Our latest gauge

a) Status OK

b) Critical event detected