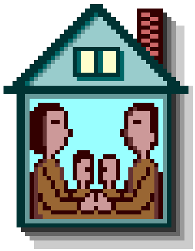
Customizing Software for Users with Special Needs

Bowen Hui
Sotirios Liaskos
John Mylopoulos

University of Toronto

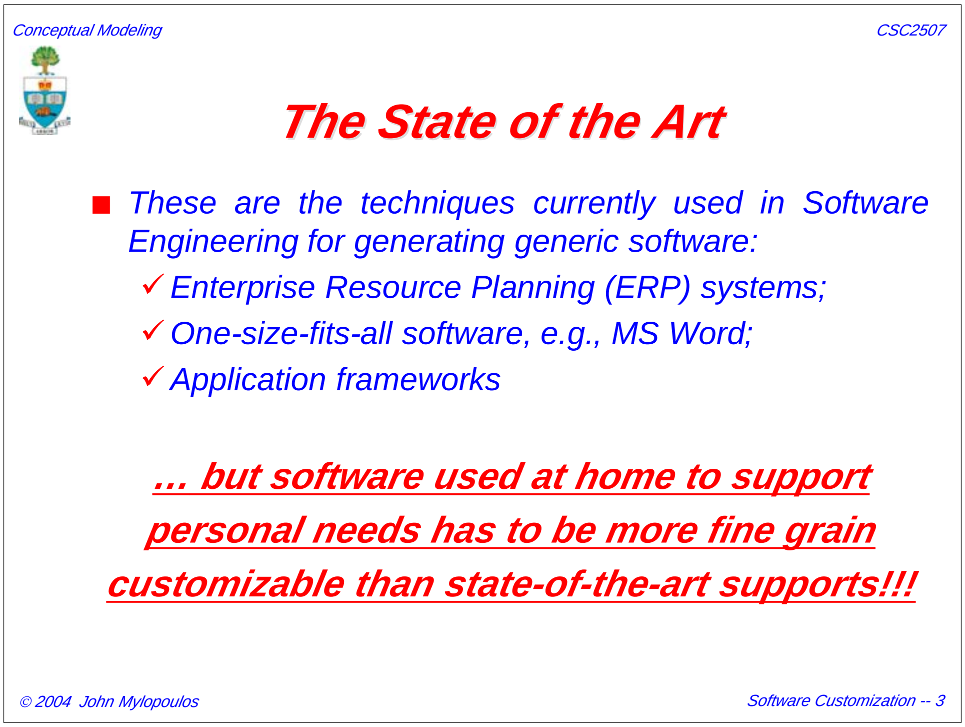
Stephen Fickas

University of Oregon



Software for Oi Polloi

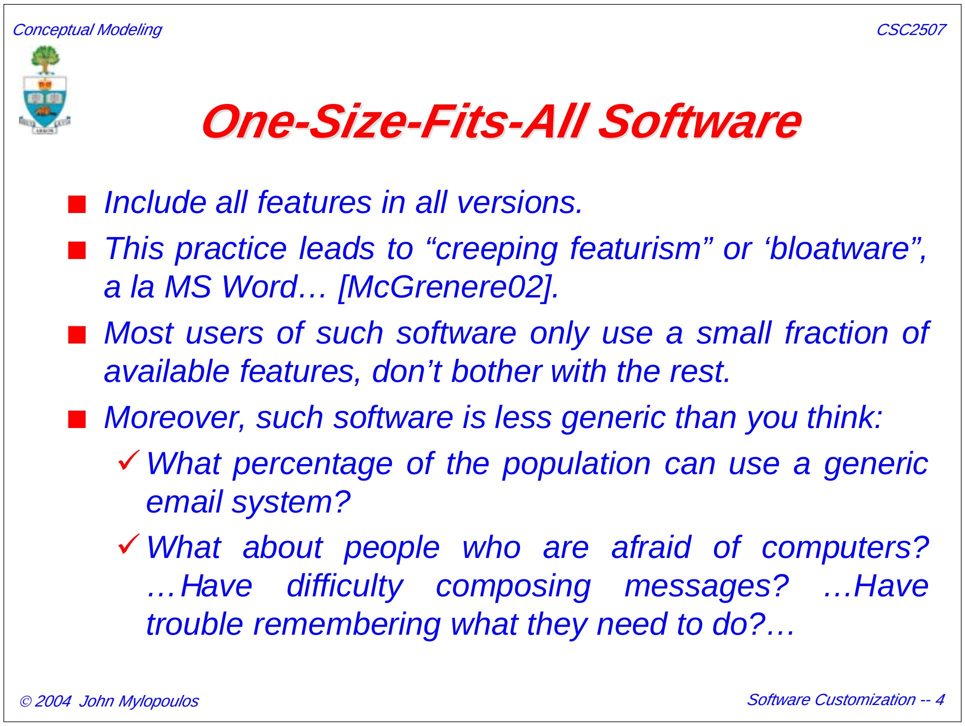
- We are interested in designing software that is usable by any member of a user community, e.g.,
  - ✓ An email system for people with a brain injury;
  - ✓ An internet system for seniors;
  - ✓ A word processor for secretaries.
- Each user in the community may have different preferences and skills. Hence the software must be ***generic***, and it must be ***customizable***.



The State of the Art

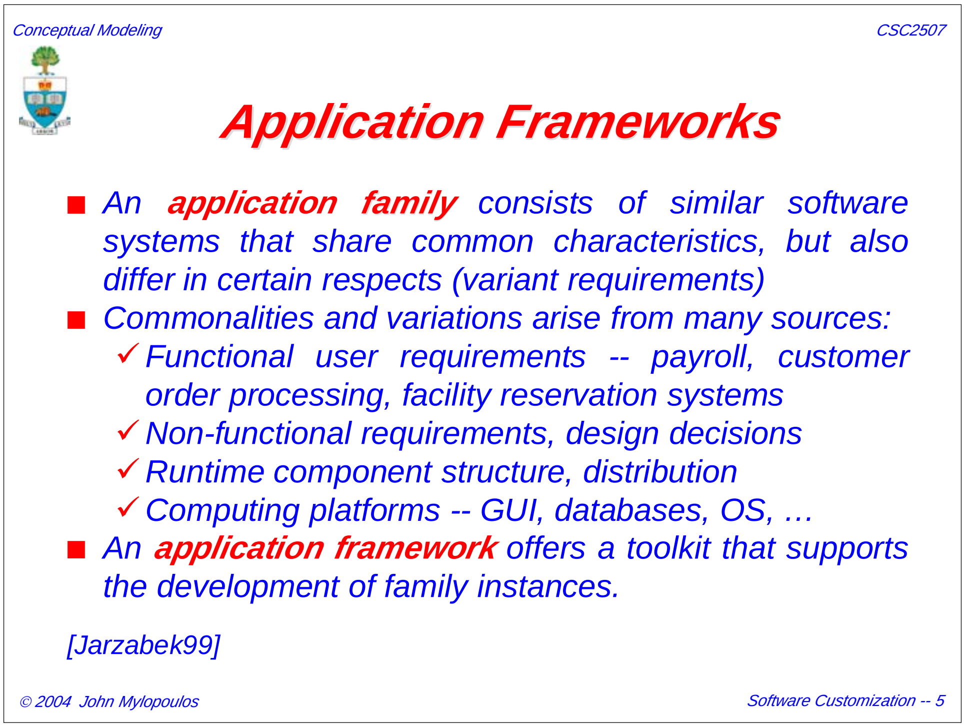
- These are the techniques currently used in Software Engineering for generating generic software:
  - ✓ Enterprise Resource Planning (ERP) systems;
  - ✓ One-size-fits-all software, e.g., MS Word;
  - ✓ Application frameworks

… but software used at home to support personal needs has to be more fine grain customizable than state-of-the-art supports!!!



One-Size-Fits-All Software

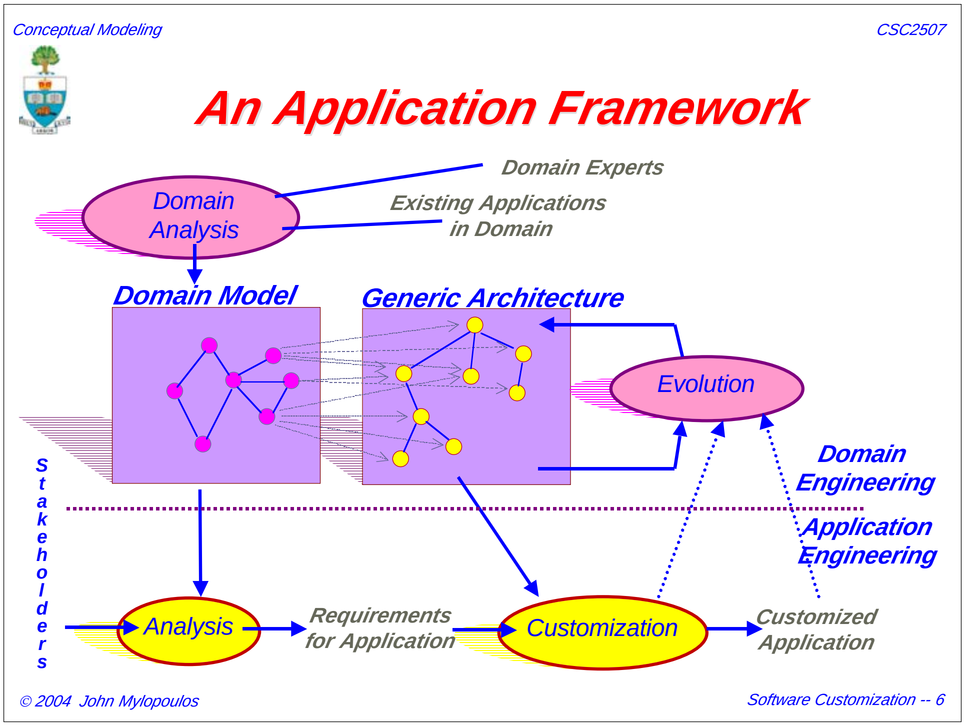
- Include all features in all versions.
- This practice leads to “creeping featurism” or ‘bloatware”, a la MS Word… [McGrenere02].
- Most users of such software only use a small fraction of available features, don’t bother with the rest.
- Moreover, such software is less generic than you think:
  - ✓ What percentage of the population can use a generic email system?
  - ✓ What about people who are afraid of computers? …Have difficulty composing messages? …Have trouble remembering what they need to do?…



Application Frameworks

- An **application family** consists of similar software systems that share common characteristics, but also differ in certain respects (variant requirements)
- Commonalities and variations arise from many sources:
  - ✓ Functional user requirements -- payroll, customer order processing, facility reservation systems
  - ✓ Non-functional requirements, design decisions
  - ✓ Runtime component structure, distribution
  - ✓ Computing platforms -- GUI, databases, OS, …
- An **application framework** offers a toolkit that supports the development of family instances.

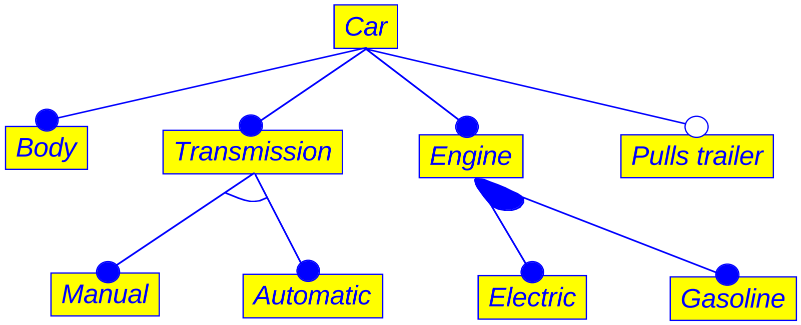
[Jarzabek99]



An Application Framework

Representing Software Variability

- *Software variability characterizes the “customization space” of a generic software system.*
- *Software variability is represented in terms of an AND/OR graph of **features.***



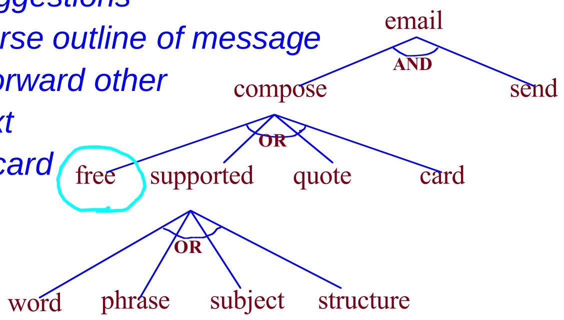
If Features are the Answer, What was the Question??

- *Features are **solution/design-oriented** elements, rather than **problem/requirements-oriented** ones.*
- *Features tell you what elements to include in a customized version of a software system, rather than needs/wants.*
- *When an (experienced) sales person tries to sell you a car, they start by asking you what will you use it for, how much are you will willing to spend etc., rather than give you a long list of features to choose from.*
- *Looking at user needs/wants is particularly important if the software you are designing isn’t embedded (e.g., in your car or refrigerator) and is intended to serve a community of users (e.g., children under 10, house spouses.)*

Is eMail Software Really Generic??

Here are useful options that might make an email software system more widely usable:

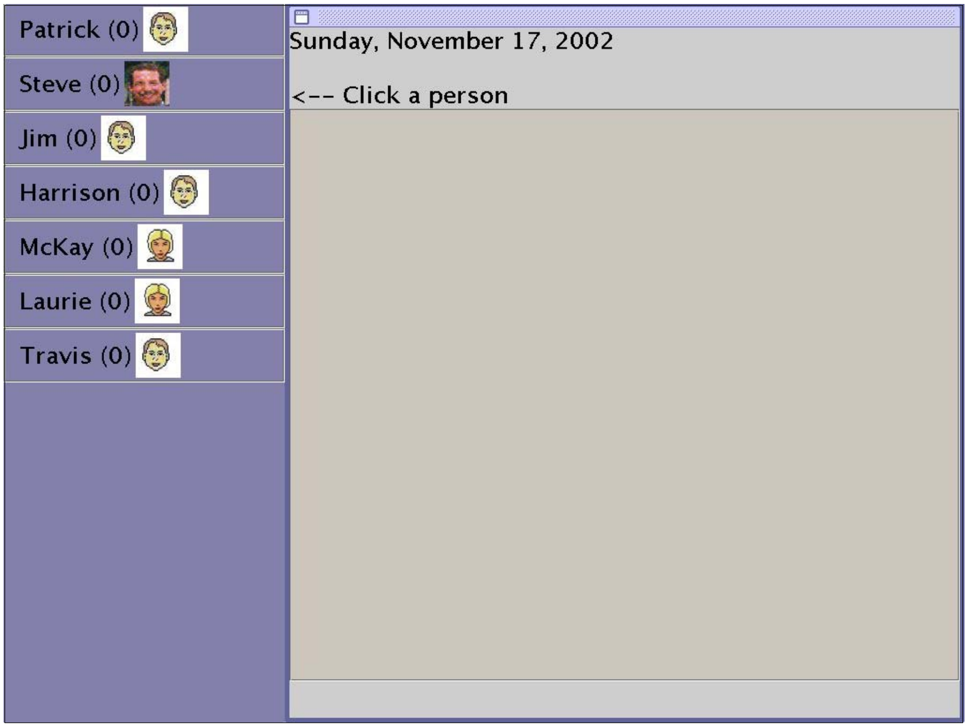
- word: word prediction
- phrase: canned phrases (fill in the blank)
- subject: topic suggestions
- structure: discourse outline of message
- quote: reply or forward other people’s text
- card: electronic card



If software is to become part of everyday life, we need to develop new techniques that will allow us to design it so that it is usable by anybody!

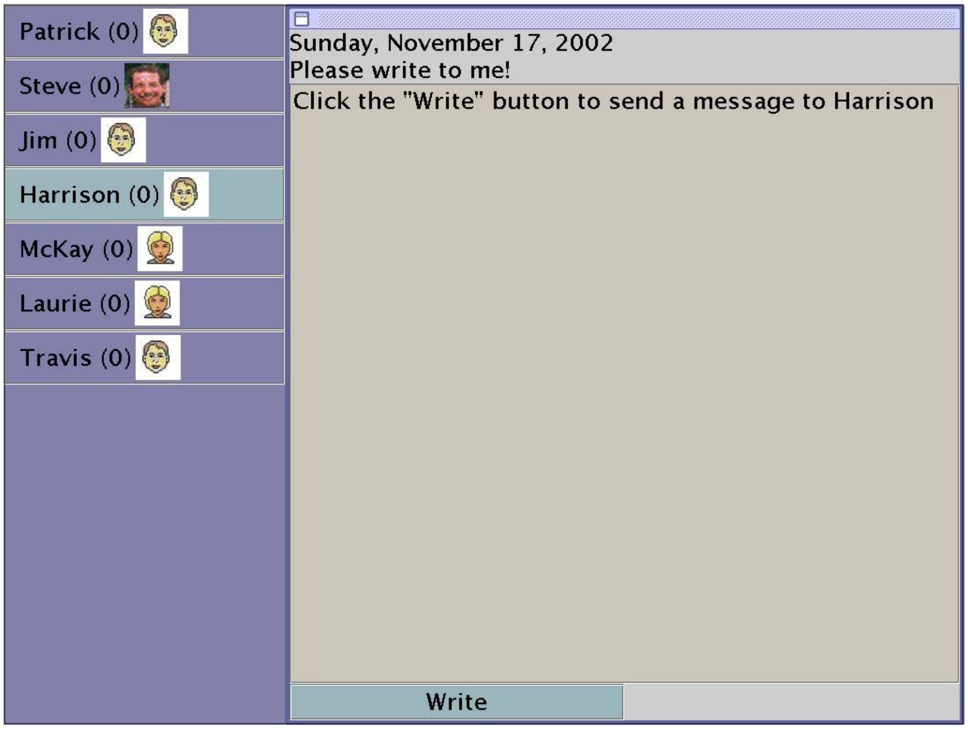
A Case Study

- Develop an email-based communication system for people with a cognitive-linguistic impairments due to a brain injury [Sohlberg02].
- These are typically younger people with very different types of deficiencies ranging from motor-control, to memory, language and initiative, to social isolation.
- [Fickas02] proposes ***personal requirements engineering*** for gathering personal requirements for a software system (e.g., email) from a potential user, also for discovering obstacles to the use of the system.
- Email was chosen for the case study because it could serve as vehicle for overcoming social isolation.
- The case study is being conducted in Oregon.



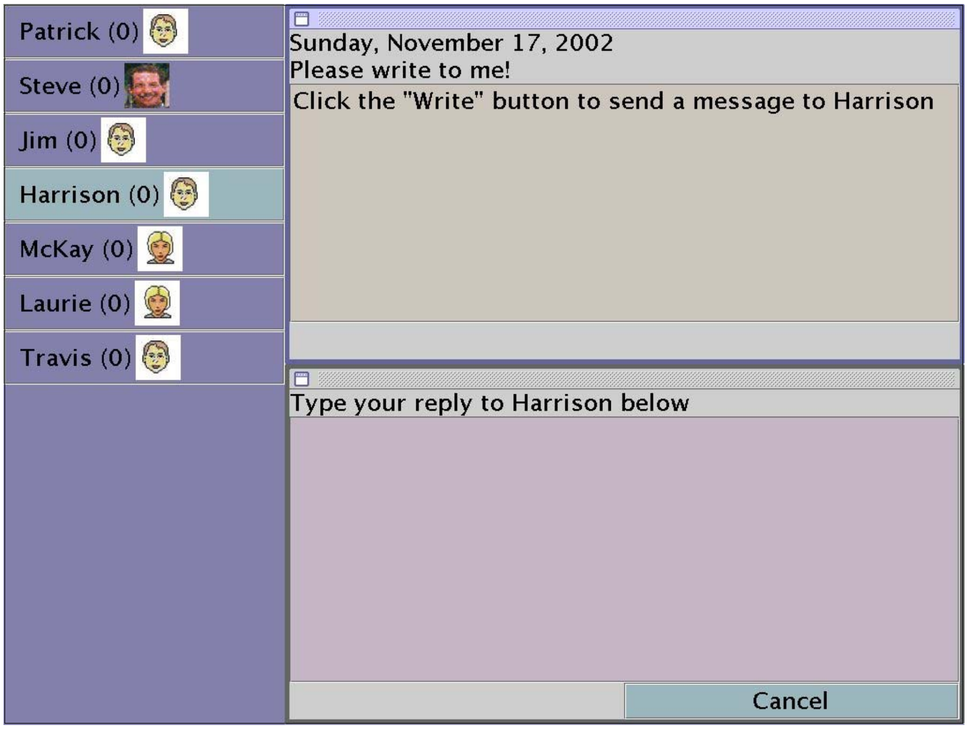
Sunday, November 17, 2002

<-- Click a person



Sunday, November 17, 2002
Please write to me!

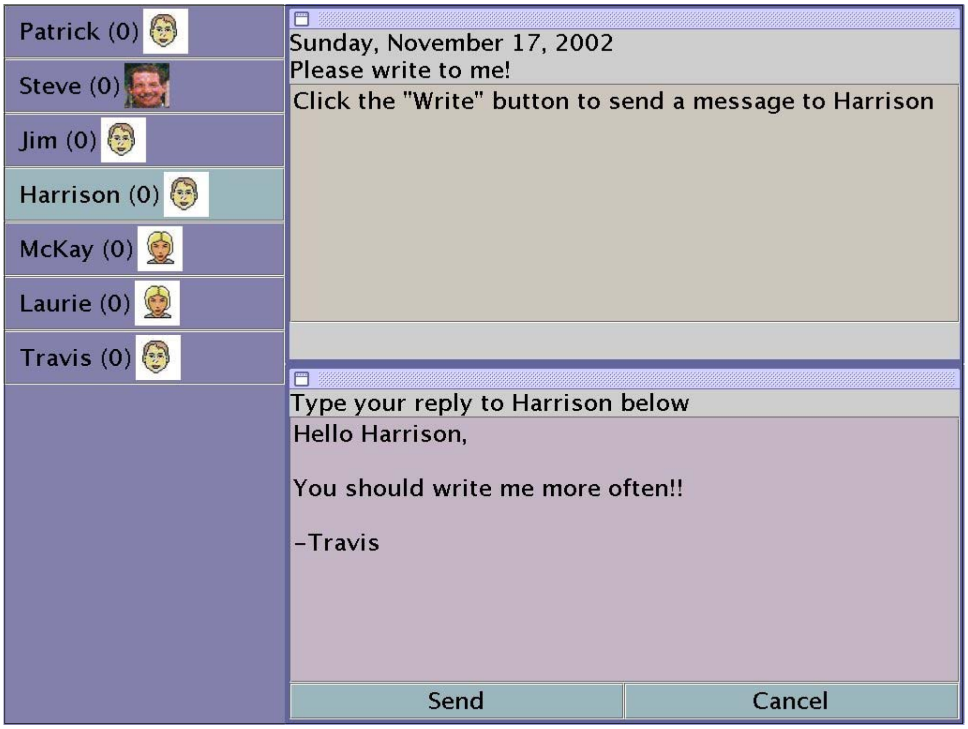
Click the "Write" button to send a message to Harrison



Sunday, November 17, 2002
Please write to me!

Click the "Write" button to send a message to Harrison

Type your reply to Harrison below



Sunday, November 17, 2002
Please write to me!

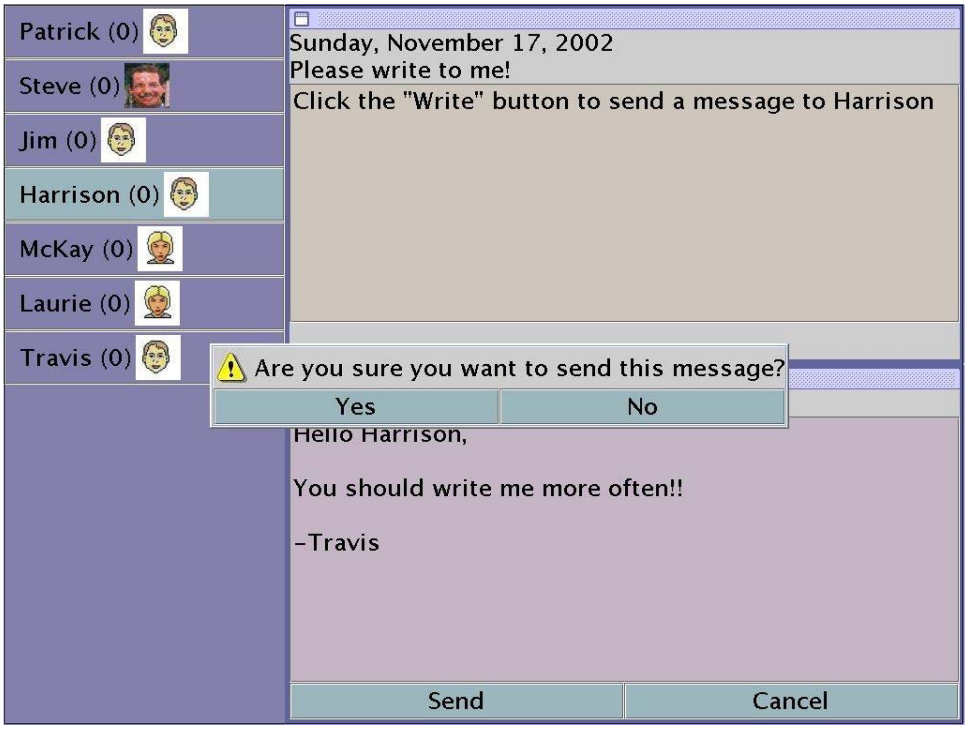
Click the "Write" button to send a message to Harrison

Type your reply to Harrison below

Hello Harrison,

You should write me more often!!

-Travis



Sunday, November 17, 2002
Please write to me!

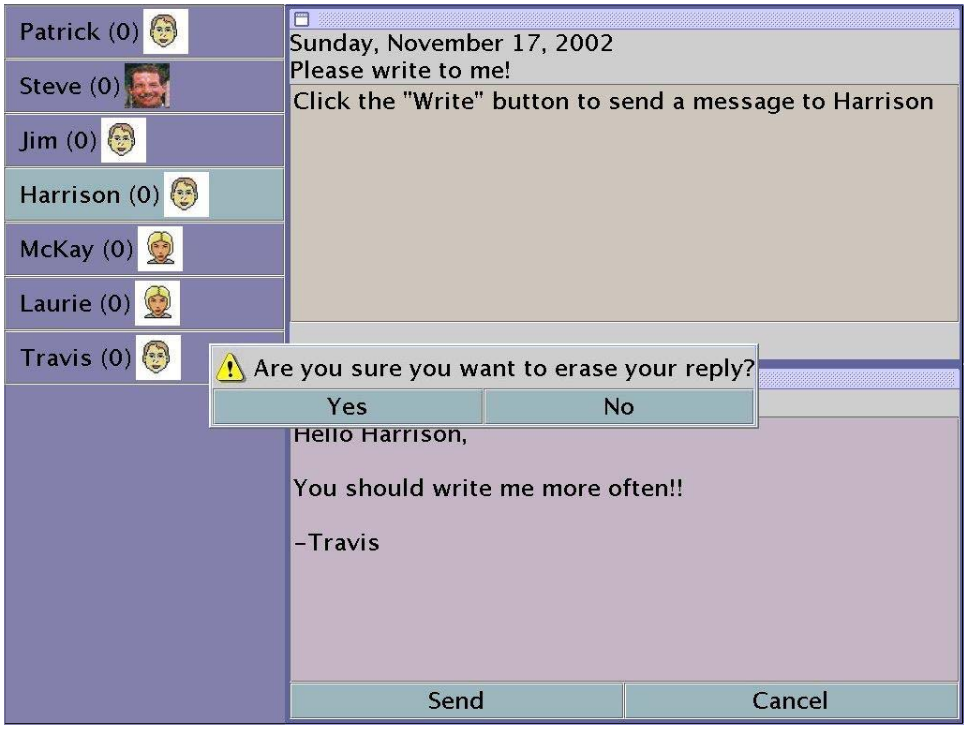
Click the "Write" button to send a message to Harrison

⚠ Are you sure you want to send this message?

Hello Harrison,

You should write me more often!!

-Travis



Sunday, November 17, 2002
Please write to me!

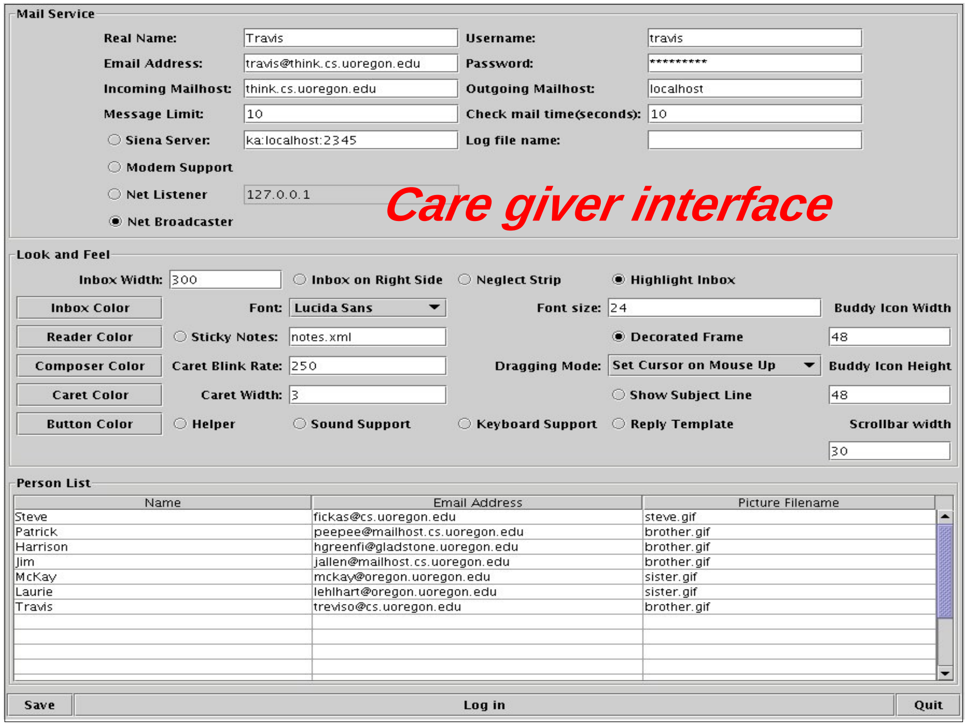
Click the "Write" button to send a message to Harrison

⚠ Are you sure you want to erase your reply?

Hello Harrison,

You should write me more often!!

-Travis



Mail Service

Real Name: Travis
Username: travis
Email Address: travis@think.cs.uoregon.edu
Password: *********
Incoming Mailhost: think.cs.uoregon.edu
Outgoing Mailhost: localhost
Message Limit: 10
Check mail time(seconds): 10
○ Siena Server: ka:localhost:2345
Log file name:
○ Modem Support
○ Net Listener 127.0.0.1
● Net Broadcaster

Care giver interface

Look and Feel

Inbox Width: 300
○ Inbox on Right Side
○ Neglect Strip
● Highlight Inbox
Font: Lucida Sans
Font size: 24
Buddy Icon Width 48
○ Sticky Notes: notes.xml
● Decorated Frame
Caret Blink Rate: 250
Dragging Mode: Set Cursor on Mouse Up
Buddy Icon Height 48
Caret Width: 3
○ Show Subject Line
○ Helper
○ Sound Support
○ Keyboard Support
○ Reply Template
Scrollbar width 30

Person List

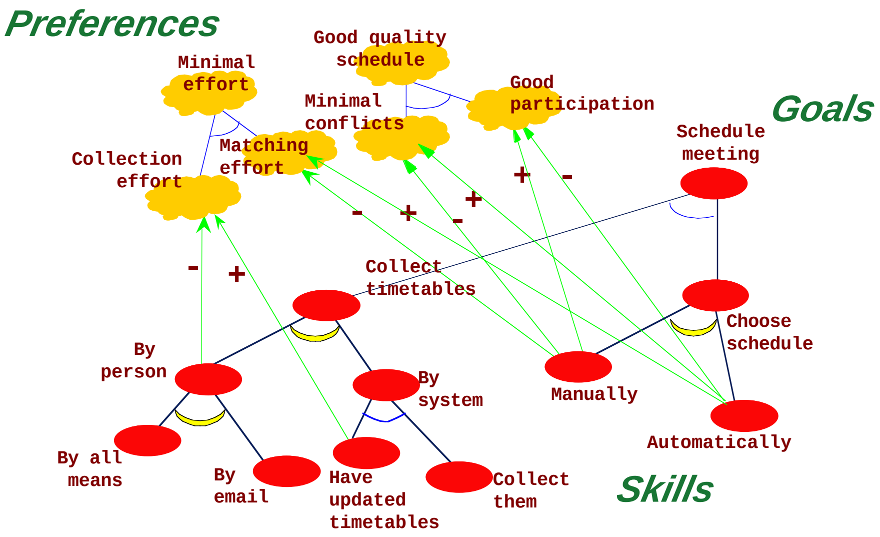
Name	Email Address	Picture Filename
Steve	fickas@cs.uoregon.edu	steve.gif
Patrick	peepee@mailhost.cs.uoregon.edu	brother.gif
Harrison	hgreenfi@gladstone.uoregon.edu	brother.gif
Jim	jallen@mailhost.cs.uoregon.edu	brother.gif
McKay	mckay@oregon.uoregon.edu	sister.gif
Laurie	lehlhart@oregon.uoregon.edu	sister.gif
Travis	treviso@cs.uoregon.edu	brother.gif

Goals, Skills and Preferences

- Gather **requirements** for the generic software system. Represent these as goals. The variability space is the set of all possible ways one can satisfy these goals. Each alternative assigns **tasks** to users of the system.
- Identify required **skills** for each task needed for the fulfillment of a goal. Disallow alternatives that assign tasks to users who don't have the necessary skills.
- Represent user **preferences** as softgoals and use them to prioritize among alternatives.
- **Customization** is defined as a mapping

  Cust: G x S x P --> V

Goals, Skills and Preferences



Goal Model Analysis

Using goal models, we can answer questions such as:

- *What is the space of alternatives supported by the generic design? …for our example, 6;*
- *Rank alternatives with respect to a softgoal:*

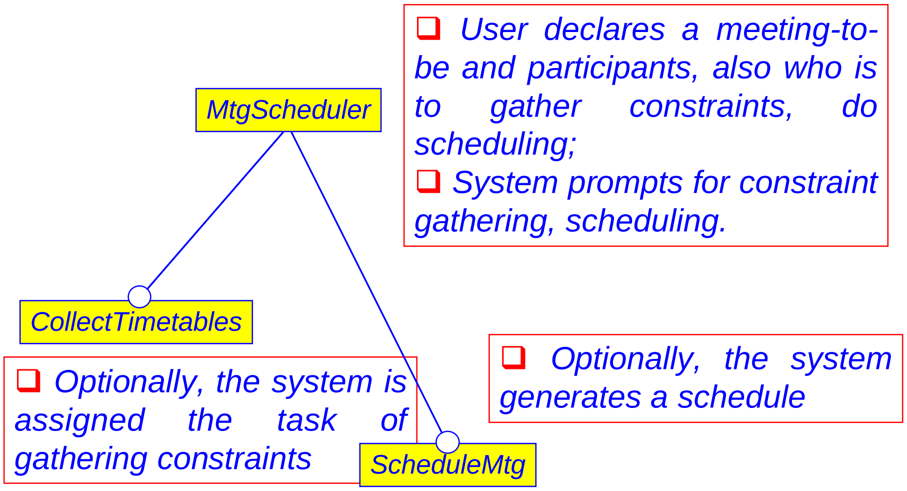
  Alternative A: *system collects timetable constraints and schedules the meeting*

  Alternative B: *people do these tasks*

  A is better than B with respect to "Minimal effort";
- *Given a goal, find all alternatives that do/don't require certain skills.*

To support these types of analysis, we need formal models of goals, skills and preferences.

Feature Model for Meeting Scheduler



❑ *User declares a meeting-to-be and participants, also who is to gather constraints, do scheduling;*
❑ *System prompts for constraint gathering, scheduling.*

❑ *Optionally, the system is assigned the task of gathering constraints*

❑ *Optionally, the system generates a schedule*

Skills Analysis

- Step 1: skills to tasks

Skills

		S1	S2	S3	S4	S5
A	T1	Yes				
	T2		Yes			Yes
B	T3			Yes		
C	T4				Yes	Yes
	T1	Yes				

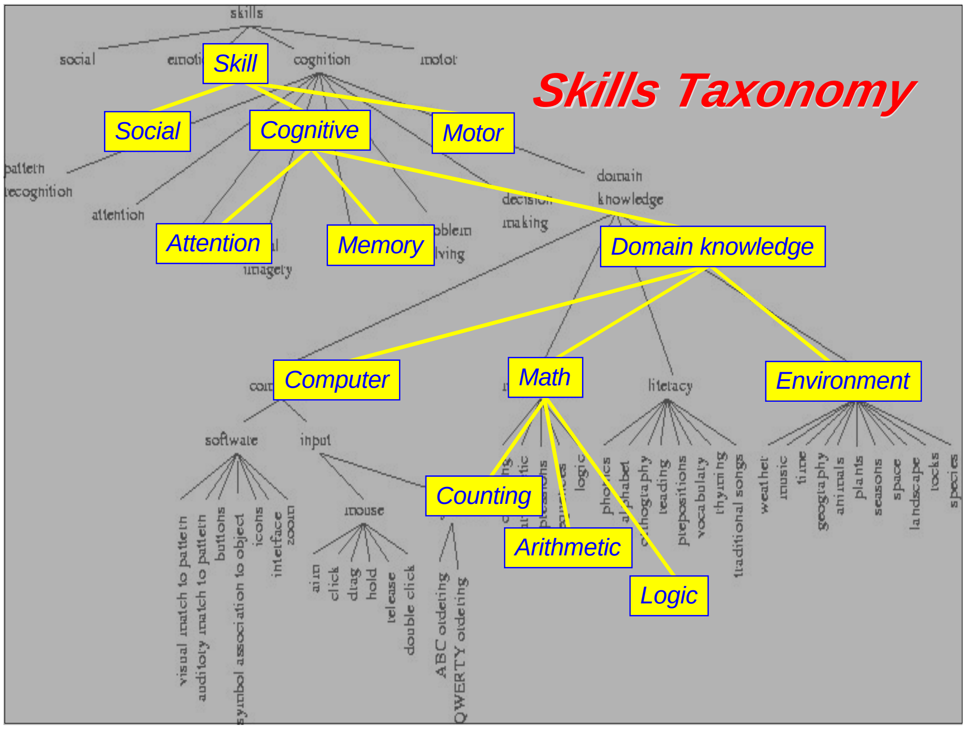
Tasks

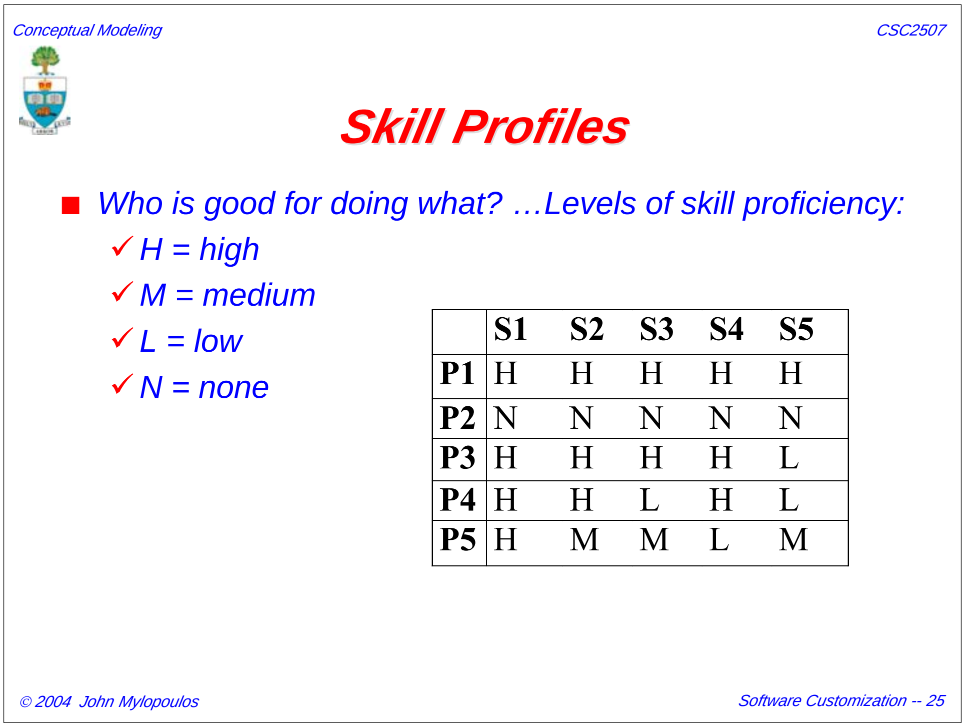
Alternatives

- Step 2: reduction

	S1	S2	S3	S4	S5
A	Yes	Yes			Yes
B			Yes		
C	Yes			Yes	Yes

Skills Taxonomy

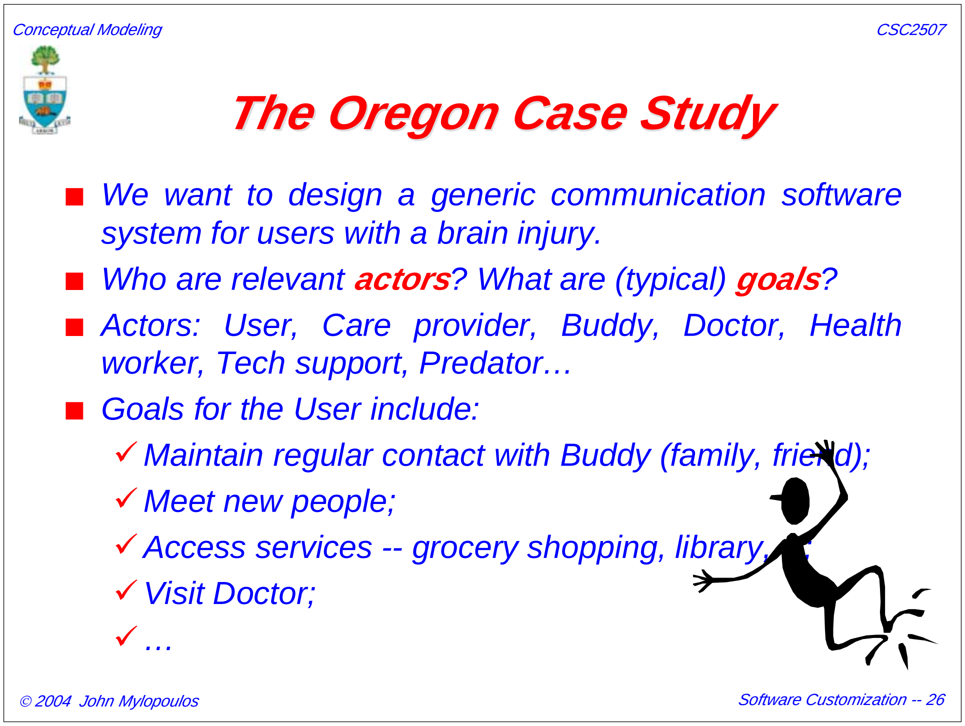




Skill Profiles

- Who is good for doing what? …Levels of skill proficiency:
  - ✓ H = high
  - ✓ M = medium
  - ✓ L = low
  - ✓ N = none

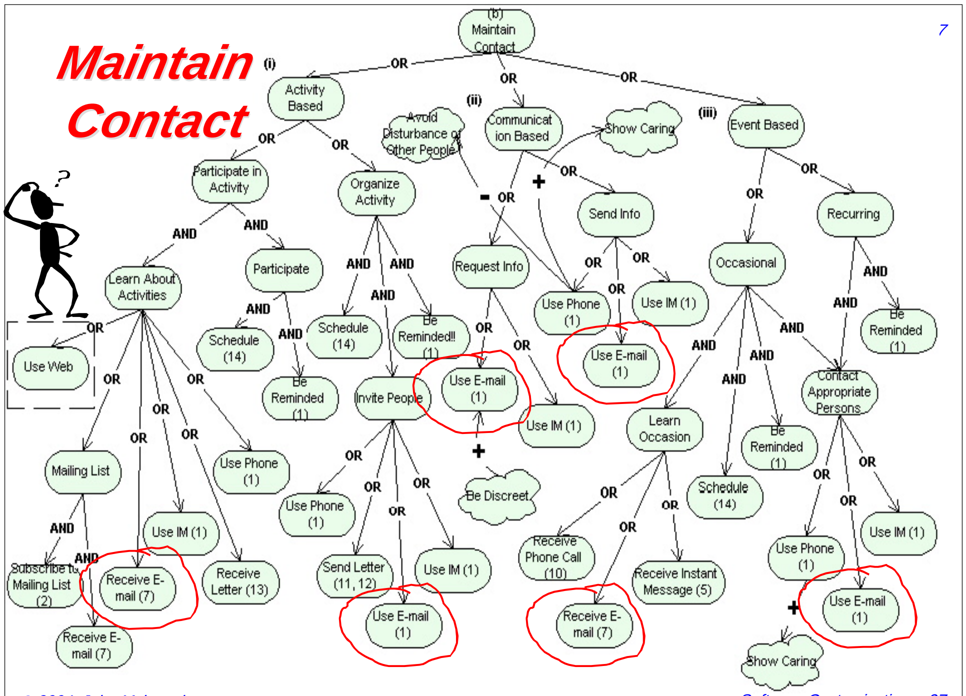
	S1	S2	S3	S4	S5
P1	H	H	H	H	H
P2	N	N	N	N	N
P3	H	H	H	H	L
P4	H	H	L	H	L
P5	H	M	M	L	M



The Oregon Case Study

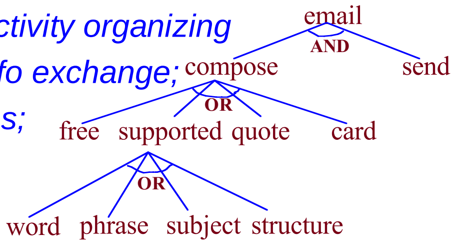
- We want to design a generic communication software system for users with a brain injury.
- Who are relevant **actors**? What are (typical) **goals**?
- Actors: User, Care provider, Buddy, Doctor, Health worker, Tech support, Predator…
- Goals for the User include:
  - ✓ Maintain regular contact with Buddy (family, friend);
  - ✓ Meet new people;
  - ✓ Access services -- grocery shopping, library, …
  - ✓ Visit Doctor;
  - ✓ …

Maintain Contact



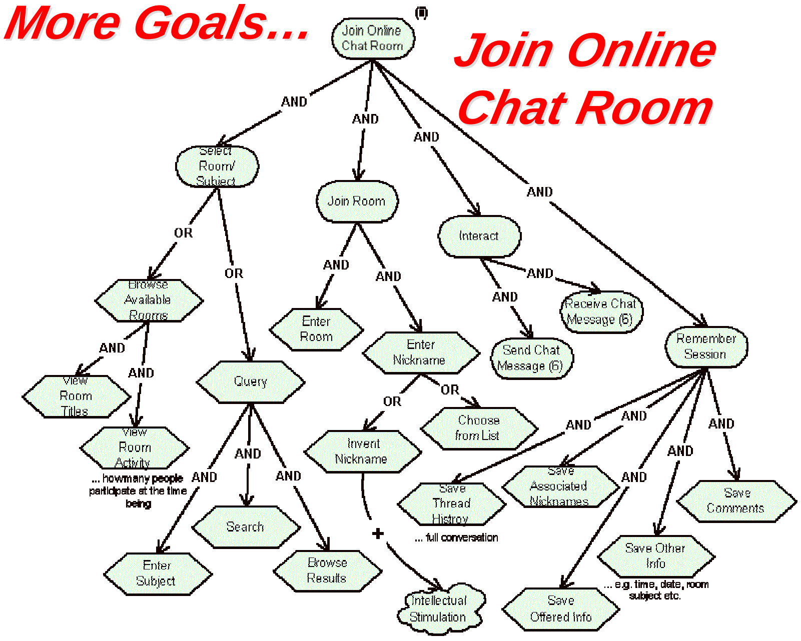
Using eMail to Maintain Contact

- Email is used in several different contexts for a user to maintain contact with buddies, e.g., to find out about activities, organize activities, exchange information, or send best wishes.
- Different contexts call for different ways of achieving the “Use email” goal:
  - ✓ Phrase support good for activity organizing
  - ✓ Subject support best for info exchange;
  - ✓ Card is best for best wishes;
  - ✓ …

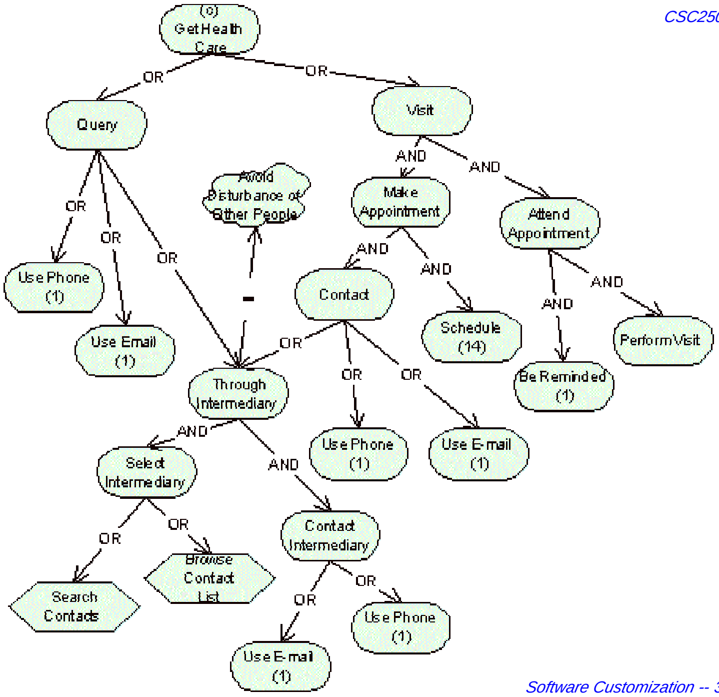


…More Goals…

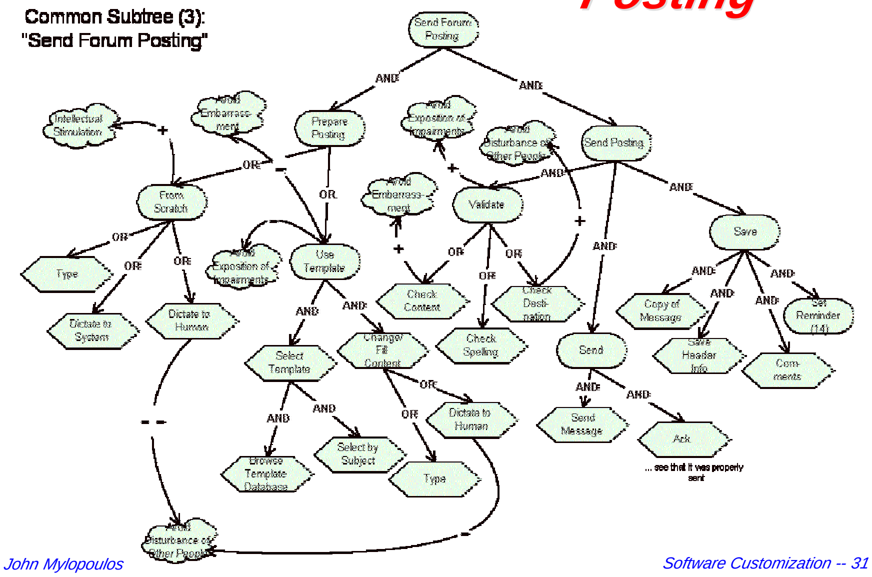
Join Online Chat Room



Get Health Care



Send Forum Posting



Goal Models

- Our current goal model (User actor only) for the Oregon case study includes about 350 goals and 400 tasks.
- The variability space is $O(10^{10})$!
- When we are done, we expect to have a generic architecture which includes hundreds of components and can support a huge number of possible customizations.
- The size of the variability space underscores the need for tools that generate and rank alternatives according to preferences/skills.

Customization and Adaptation

- *We customize by asking a user to choose among alternatives (**design-time** customization.) For people with special needs, an assessment may be required.*
- *Alternatively, we can make the software **adaptable** or **adaptive** through run-time monitoring mechanisms.*
- *A system is **adaptable** if it allows users to switch it from one alternative to another at run-time. A system is **adaptive** if it switches automatically at run-time from one alternative to another by using some form of machine learning.*
- *We'd like to support all three forms of tailoring a generic software system to the needs of a user.*

"…Technology can play a role helping the cognitively impaired lead more independent lives…"

The Toronto Star,

March 24, 2003

Outlook

- We don’t assume that this kind of software system will be used via a conventional computer, i.e., box, monitor and keyboard.
- Instead, the idea is to have it run on a “smart home” infrastructure which involves devices that participate in a wireless LAN (phones, oven, frig, sensors, TV, CD player,…), with an operating system running on top.
- Input modes include voice and touch screens, while output modes include voice and activators.
- Industry is already working on the infrastructure for this (hardware and software.)
- We are developing techniques for designing the software that runs on top.

Conclusions

- We have outlined a framework for designing generic software, founded on the concepts of *goal* and *actor*.
- The framework can be used to design software for people with special needs. It could also be used to design software for a community whose members share common goals, e.g., internet-based services for:
  - ✓ High school students, with a focus on finding course material, chatting with friends, playing games;
  - ✓ House spouses, with an emphasis on cooking information, finding bargains, chatting with friends, entertainment;
  - ✓ Senior citizens, with an emphasis on health information and services, chatting with friends;
  - ✓ …

References

- [Antón96] A. I. Anton, "Goal-based Requirements Analysis." Proceedings 2nd IEEE International Conference on Requirements Engineering, April 1996.
- [Boehm96] B. Boehm, H. In, “Identifying Quality-Requirement Conflicts,” IEEE Software, March 1996, pp. 25-35.
- [Cheong99] Cheong, Y.C. and Jarzabek, S. “Frame-based Method for Customizing Generic Software Architectures," Proceedings. Symposium on Software Reusability, (SSR’99), Los Angeles, May 1999, 103-112.
- [Chung00] L. Chung, B. Nixon, E. Yu, and Mylopoulos, J., Non-Functional Requirements in Software Engineering, Kluwer Publishing, 2000.
- [Dardenne93] A. Dardenne, A. van Lamsweerde and S. Fickas, “Goal-Directed Requirements Acquisition”, Science of Computer Programming, 20, pp. 3-50, 1993.
- [erpfans.com] www.erpfans.com.
- [Fickas02] Fickas, S., Ehlhardt, L., Sohlberg, M., Todis, B., Personal Requirements Engineering, TR-45-02, Dept. of Computer Science, University of Oregon, October 2002.
- [Forrester98] Forrester Report, Conquering Customization, March 1998.
- [Forrester99] Forrester Report, ERP eCommerce Realities, April 1999.
- [Glass99] Glass, R., Vessey, J., “Enterprise Resource Planning Systems: Can they Handle the Enhancement Changes Most Enterprises Require?”, Proceedings EMRPS’99, Venice, November 1999.

References

- [Jarzabek99] Jarzabek, S., :”Synergy between Component-Based Software Engineering and Generative Approaches to System Families”, Software Engineering seminar, University of Toronto, October 1999.
- [Macala96] Macala R., Stuckey, L. Jr. and Gross, D. “Managing Domain-Specific, Product-Line Development,” IEEE Software, May 1996, 57-67.
- [McGrenere02] McGrenere, J., Baecker, R., Booth, K., “An Evaluation of a Multiple Interface Design Solution for Bloated Software”, Proceedings CHI’02, Minneapolis, April 2002.
- [Mylopoulos99] J. Mylopoulos, L. Chung and E. Yu, “From Object-Oriented to Goal-Oriented Requirements Analysis,” Communications of the ACM, 42(1), pp. 31-37. January 1999.
- [Sohlberg02] Sohlberg, M., Ehlhardt, L., Fickas, S., Todis, B.,CORE: Comprehensive Overview of Requisite Email Skills, TR-02-01, Dept. of Computer Science, University of Oregon, October 2002.
- [van Lamsweerde95] A. van Lamsweerde, R. Darimont, and P. Massonet, “Goal Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt.,” Proceedings 2nd IEEE International Symposium on Requirements Engineering, York, UK, March 1995, 194-203.
- [Parnas76] Parnas, L., “On the Design and Development of Program Families”, IEEE Transactions on Software Engineering 2(1), January 1976.
- [Rolland99] Rolland, C., “A Goal-Driven Approach to Modeling COTS Acquisition Impacts”, Proceedings EMRPS’99, Venice, November 1999.
- [sei.cmu.edu] www.sei.cmu.edu/domain-engineering