Dynamic Models -

Document driven information system for policy implementation

A case study in developing curriculum regulation and conformity using Protege

Dr. Mike Hobbs & Dominic Myers
Department of Computer Science,
Anglia Polytechnic University
m.h.w.hobbs@apu.ac.uk
Main points of the talk

- The Context - using knowledge to improve information systems.
- Dynamic document driven models - from paper to XML
- The application domain - module choice for higher level degree programs.
- The strategy -
  Documents + Mark-up + Ontology + Agents = Dynamic models
- The role of Protégé
- Evaluation criteria
Context

- Typical communication of decisions
  - Top down
  - Committee led
  - Recorded in minutes and memos

- Better communication / information support
  - Check decisions against policy knowledge base.
  - Facilitate changes to policy knowledge base.
  - Verify choices / implementation of rules.
  - Disseminate relevant information
Why Dynamic Documents?

- **Benefits**
  - Organisations are mostly document based
  - Tap into existing ways of working
  - Systems can run in parallel

- **Requirements**
  - Need to have ‘easy’ mark-up
  - Need to validate input
  - Need to interpret / disseminate information when and where required - e.g. software agents.
Application Domain

Module Choice

Different degree programs are made up with different combinations and of modules, with many options.

Modules have pre-requisites, co-requisites and non-requisites.

Modules can run or be taken in different semesters.

Students need to make choices for options and timings based on moderately complex rules.

Currently limited system support or validation.
Features of the Application Domain

- The case study has:
  - Clear rules
  - Small scale - with clear boundaries
  - Modest amount of change
  - Some complexity
  - Developers with good understanding of current system
  - Evaluation against existing system
Questions we hope to answer:

- Representation - can a Protégé ontology represent the domain?
- Provide ‘proof of concept’ end to end - from input mark-up of documents to delivery of information to students.
- Support for change management - how much can be changed before KB system fails
- Evaluate various KB tools and editors
- Establish evaluation criteria to assess practical use
System Design

Core Components

- Documents - information source
- Mark-up - Semantic mark-up OWL
- Ontology - Providing the domain knowledge structure
- Rule base - Provides actions / functionality
- Interface - Web browser or agent based system
Domain transformations

- Document ⇒ Information resource
  - Automation - integrate or add on?

- Information resource ⇒ Ontology
  - Incorporate new information
  - Validate against existing conceptual model
  - Change management via meta ontology

- Ontology ⇒ Rule base
  - Role of ontology
  - Abstraction of concepts
- Rule base $\Rightarrow$ Interface
  - Agent based system

- Protégé Ontology creator editor
  - Good support with user/developer community
  - Stable but developing tool (via plug-ins etc.)
  - Platform for application specific tools.
Alternative model

- Classical RDBMS

- RDBMS + Interface (SQL) = information system
  - MySQL + PHP
  - Used to evaluate KBS approach.
  - Less flexible / more hands on?
Conclusion

Case study motivated by
- Need to demonstrate advantages of KBS approach
- Desire to extend capabilities of Information systems
- Provide test bed for KBS tools and techniques
- Provide application domain for agent based systems

So far…
- Identified system requirements
- Core areas of evaluation
BUT...

- need to determine detailed model
- Assign responsibilities to different parts of the system
- Focus on ‘interesting questions."