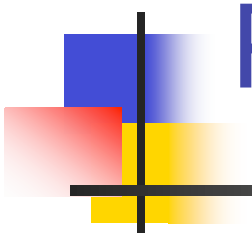
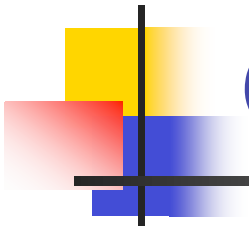
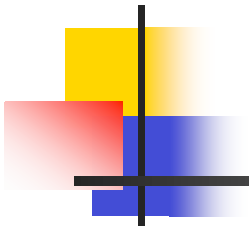


Protégé High Level Design



Outline

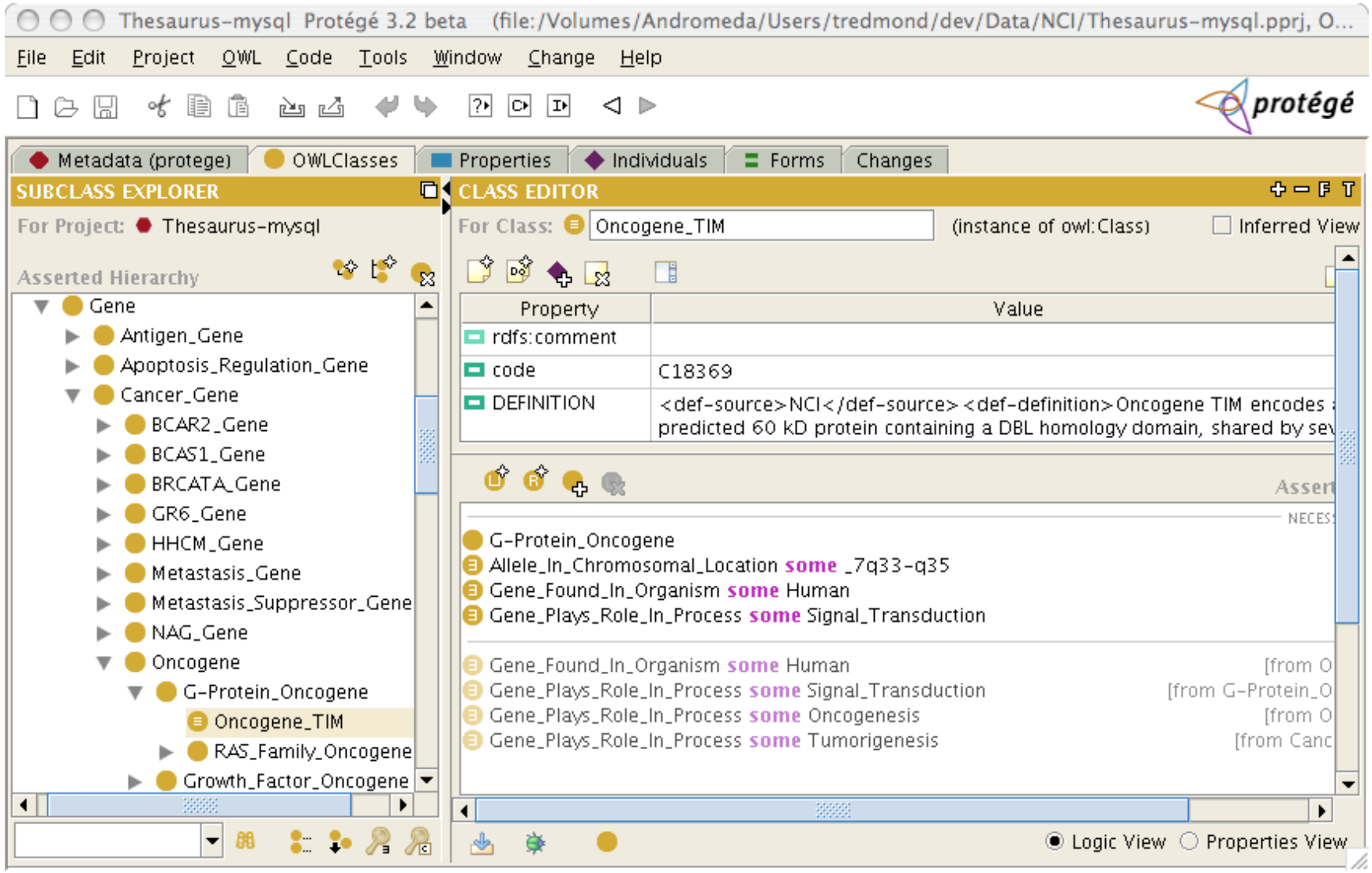
- What is Protégé
- Writing Applications on Protégé
- Protégé Design
- Recent NCI Work
- Future Directions



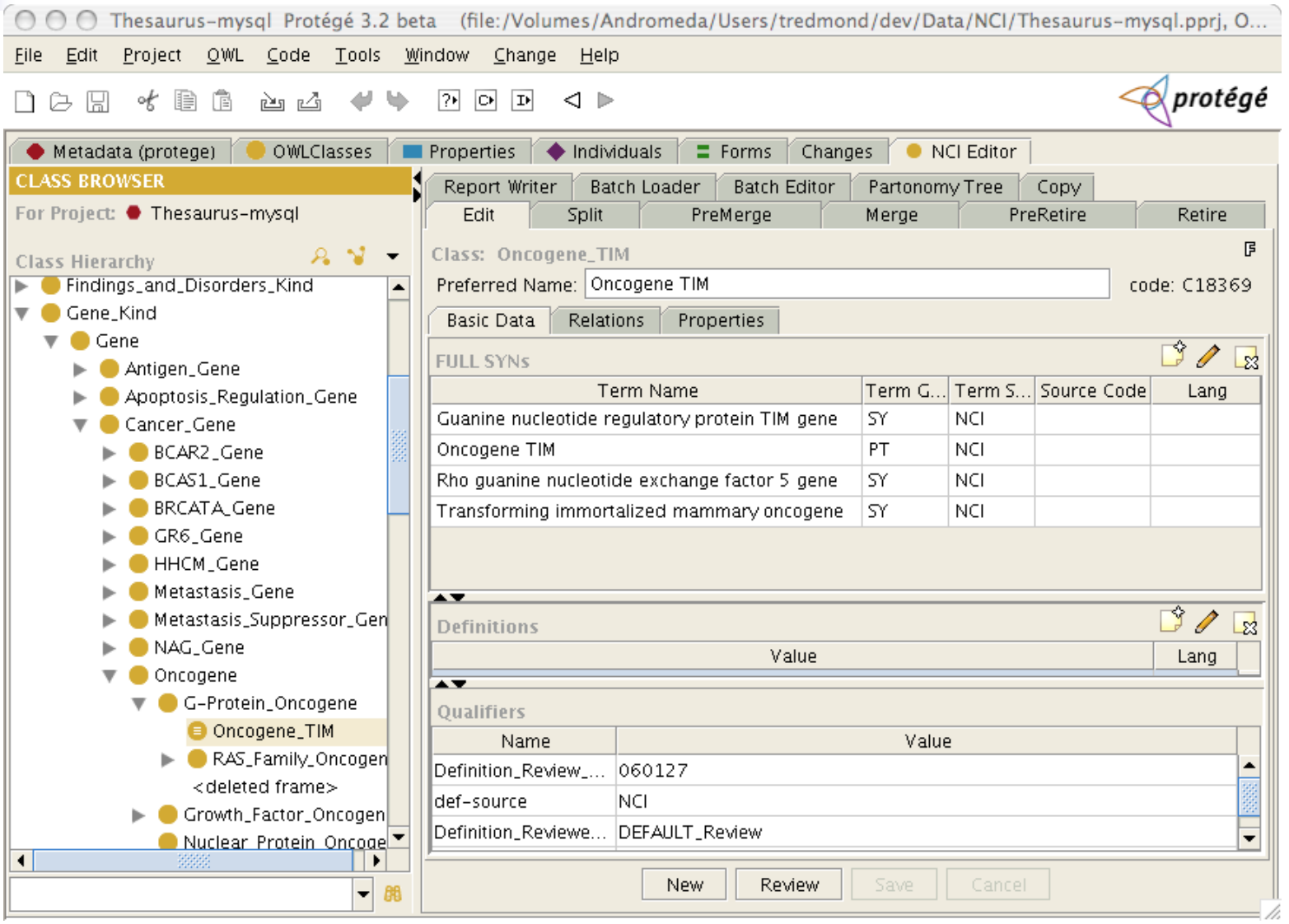
Introducing Protégé

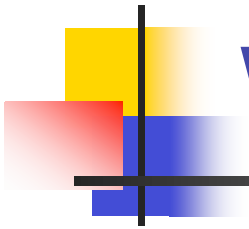
- Graphical Ontology Editor
- Pluggable Tool
  - Tab,Slot, Project, Backend, Import, Export
  - Plugin Support is basis for many applications
- Application Component

Graphical Ontology Editor



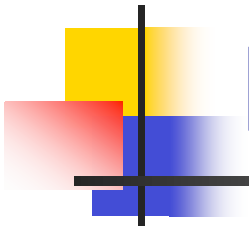
Pluggable Tool





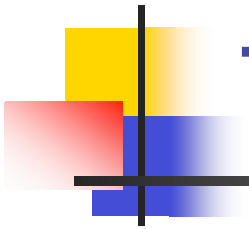
What is a Plugin?

- Extension to Protégé
  - Requires no source code modifications
  - Loaded and managed by system
  - Changes way Protégé works
- Implementation of a Java *interface*
- Packaged as *jars*
- Installed in subdirectory of Protégé *plugins*



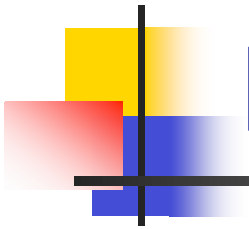
How Plugins Work

- Protégé, at startup, loads jars directly below *plugins* subdirectory
- Jars contain description of contained plugins
  - meta_inf/manifest.mf
- System creates instances of plugin
- System calls plugin methods when needed “Don’t call us, we’ll call you.”



Types of Plugins

- TabWidget
- SlotWidget
- KnowledgeBaseFactory (“Backend”)
- ProjectPlugin
- ExportPlugin
- CreateProjectPlugin

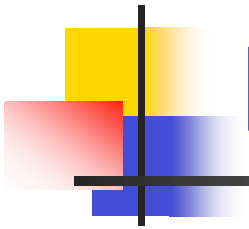


Plugin: TabWidget

- What is it?
  - Large piece of screen real-estate
  - Can interact with domain KB

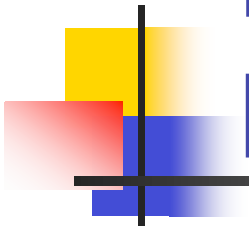
    browse, change, delete, corrupt

  NCIEdit Tab
- What are its limitations?
  - Difficult to supplement or even interact with other tabs



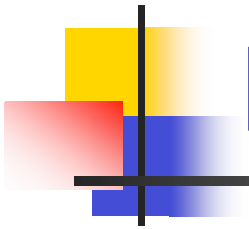
Plugin: SlotWidget

- What is it?
  - UI Control which allows the user to display and modify a slot value
  - Follows a protocol for hiding interaction KB
- What are its limitations?
  - Works best with a *single* slot



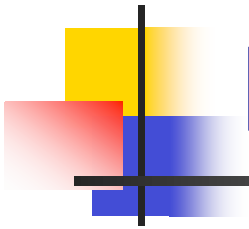
Plugin Type: KnowledgeBaseFactory

- What is it?
  - Replacement for standard storage mechanisms
    - Database
    - External server
    - …
  - Allows for parsing of different file formats
- What are its limitations?
  - Difficult to manipulate UI
  - Implementations tend to be buggy



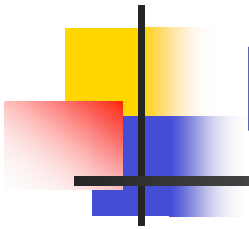
Plugin Type: ProjectPlugin

- What is it?
  - Code that executes when “things happen” to a project (create, load, display, close, etc)
  - Get access to project, view, menu bar, tool bar and can modify them as you like
- Example
  - Changes Plugin which tracks changes as they occur.



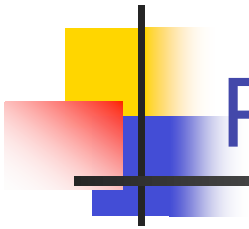
Plugin Type: ExportPlugin

- What is it?
  - Code that saves (part of) a knowledge-base in any format to *somewhere else*
    - files, servers, web, …
  - No change of the current backend
  - No guarantee of “lossless round trip”
  - No “live” connection



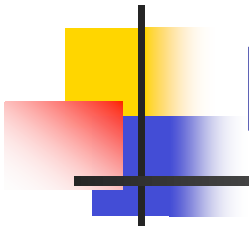
Plugin Type: ImportPlugin

- What is it?
  - Code that creates a knowledge-base from information from *somewhere else*
    - files, servers, web, …
  - No change of the current backend
  - No guarantee of “lossless round trip”
  - No “live” connection



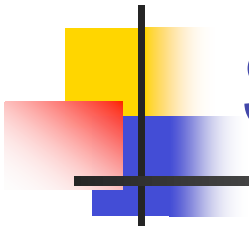
Protégé as an Application Component

- Tab Plugin (*NCI Edit Tab*)
- Standalone Application
- Distributed Application



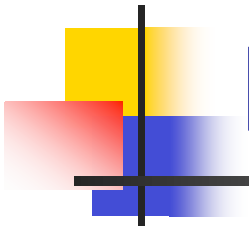
Protégé Tab as An Application

- Description
  - Create a custom tab plugin
  - Configure Protégé to just display your tab
- Pros
  - Simple
  - Great for few users
  - Iteration (change of model, data, app) is very easy
- Cons
  - Protégé must be installed
  - Difficult to permanently disable standard functions
  - Stuck with Protégé menus, toolbar, etc
  - No security on underlying model and data
  - User really should know something about Protégé



Standalone Application

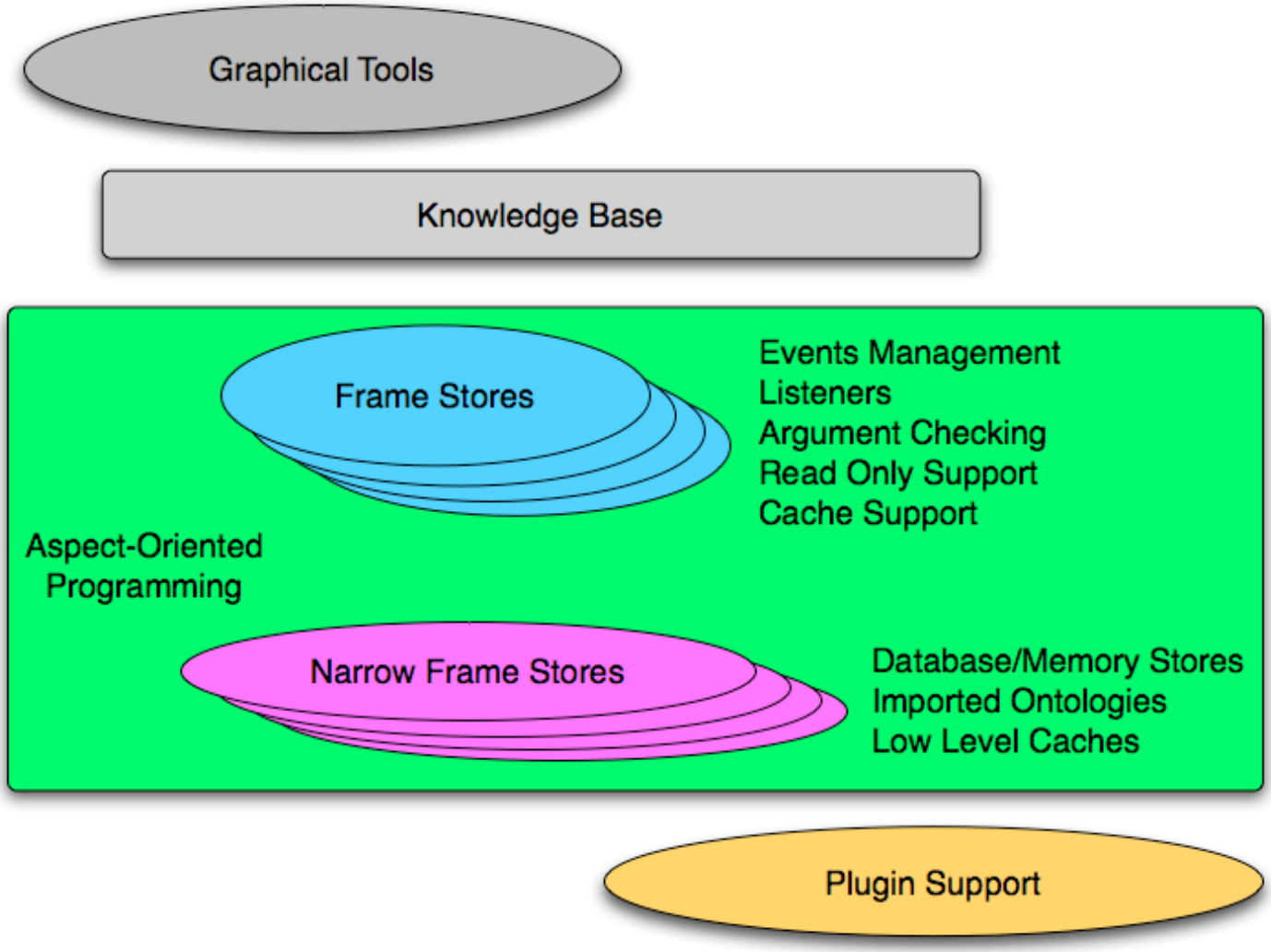
- Description
  - Write standalone Java Application
  - Call into the Protégé API for knowledge base access
  - Often evolves from a Tab
- Pros
  - No need to install Protégé
  - User doesn’t need to know anything about Protégé
  - Underlying model and data are as secure as you want
  - Can use some or none of the Protégé UI, as desired
    - Forms for classes and instances are available
    - Some tabs will work
- Cons
  - Iteration somewhat more difficult than as Tab



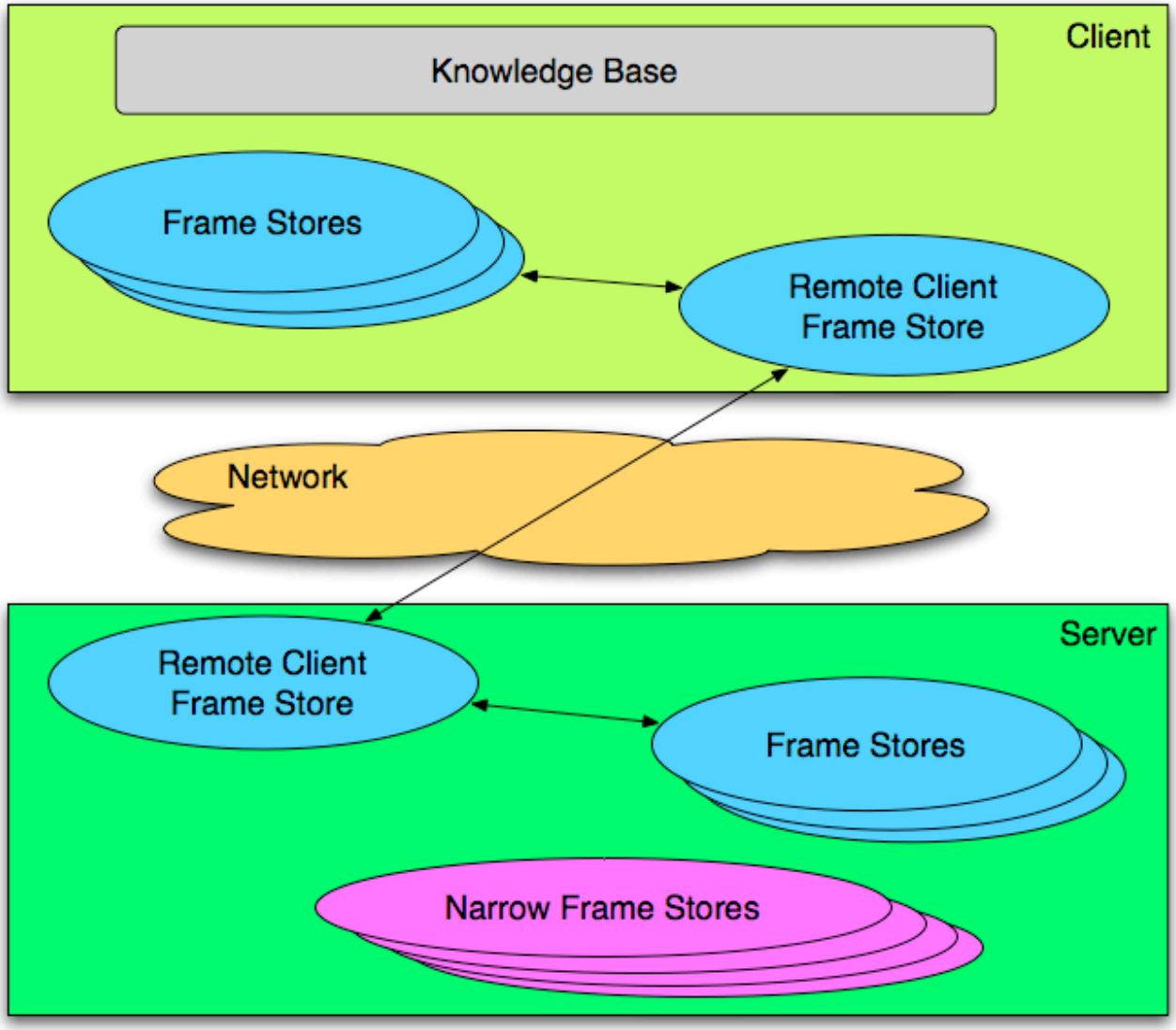
Network Solutions

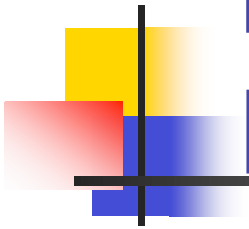
- Applets
- Java WebStart
- Servlets and Java Server Pages
- *Protégé RMI server*
- Custom server

Protégé Architecture



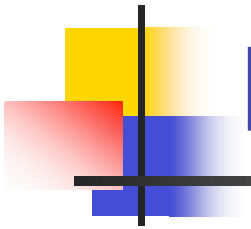
Server-Client Architecture





Protégé Performance Enhancements

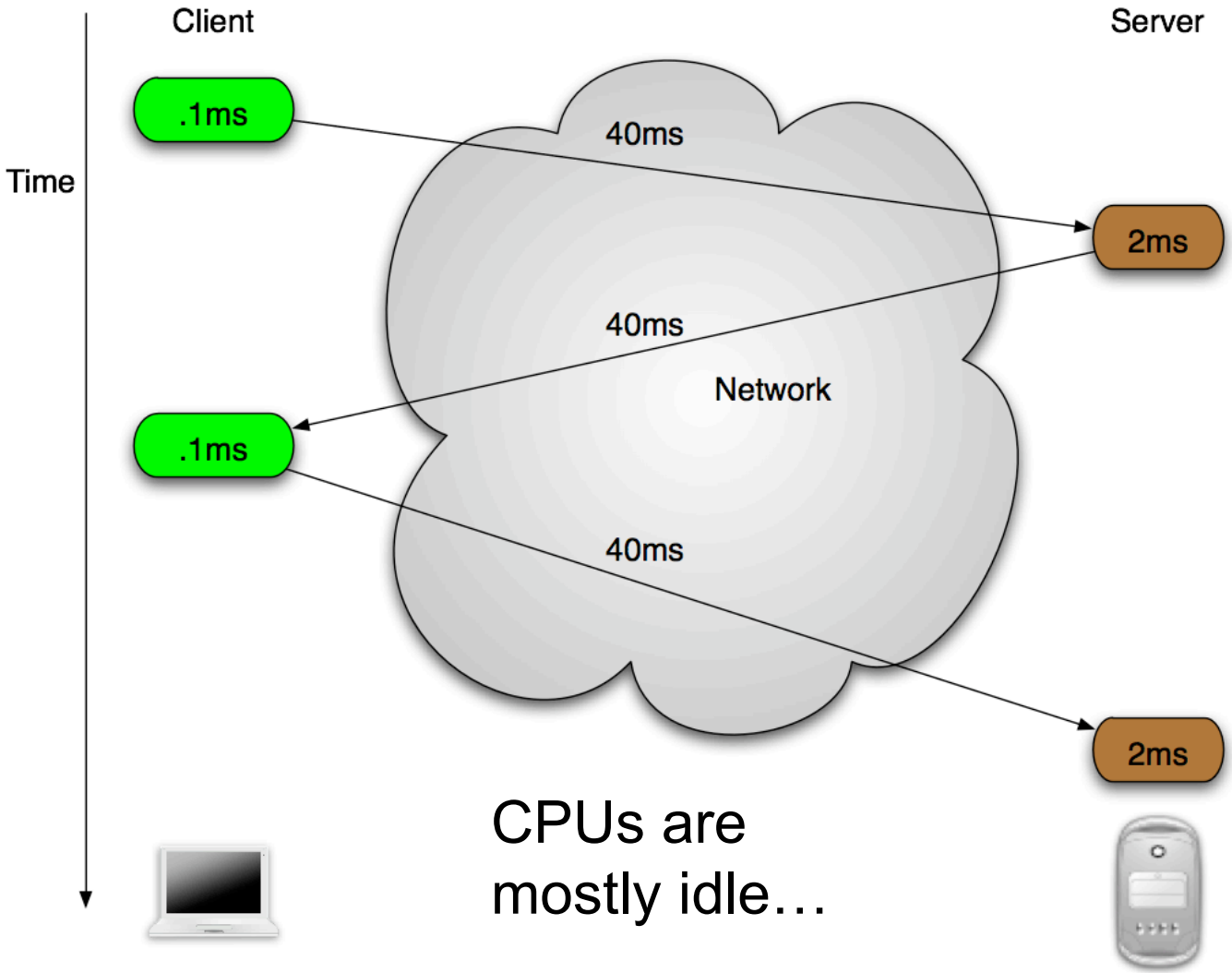
- Slow Performance When
  - Protégé Server-Client
  - Database Backend
  - OWL
  - Complex Ontology
- Focus on Server-Client and OWL



Main Issues

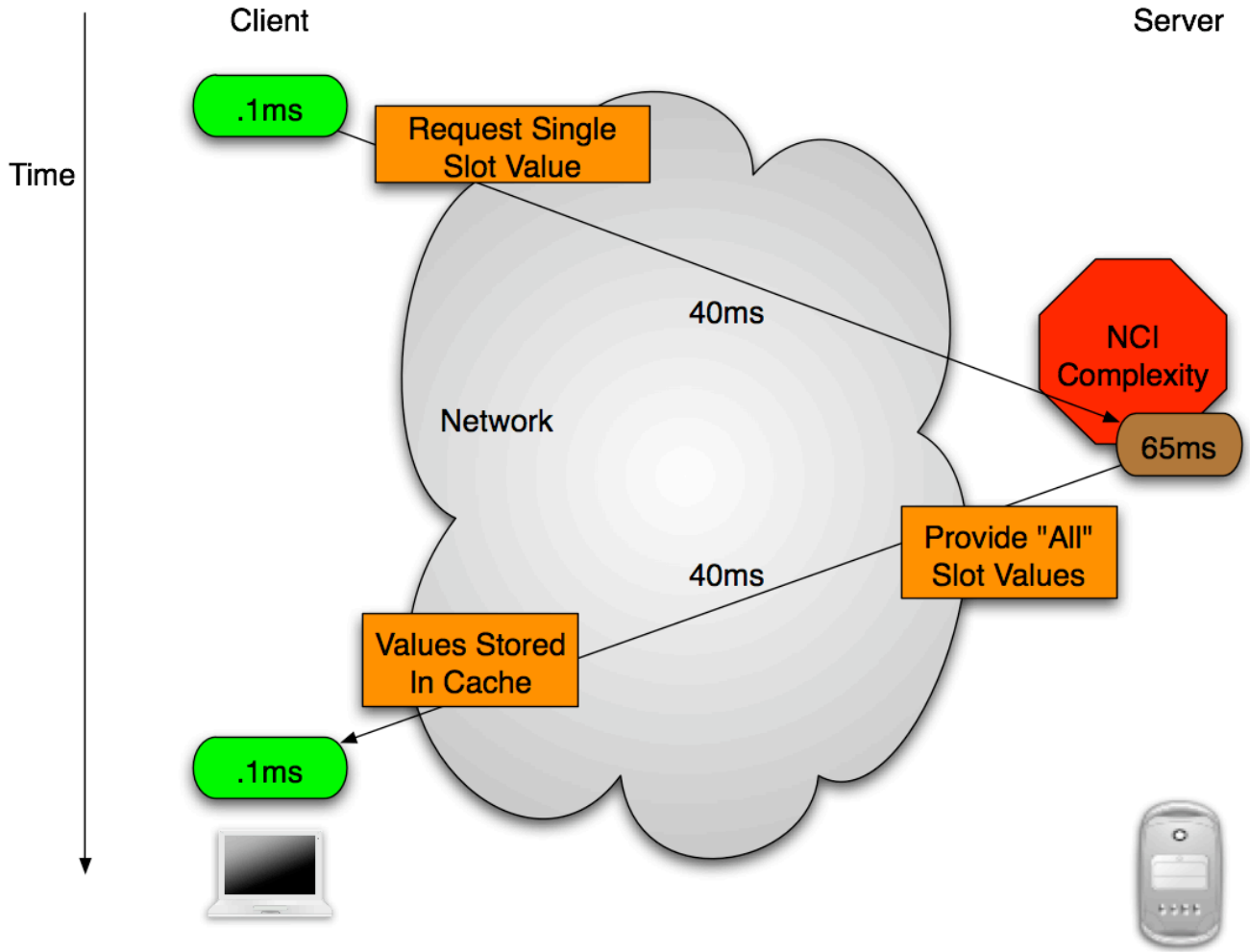
- Protégé Frames
  - Granularity of server locks
  - Caches were slow
  - Caches were incorrect
  - Role of Transactions
- Protégé OWL
  - Inefficient code
  - Design Decisions
  - Role of Inference

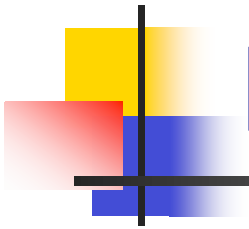
Basic Server-Client Problem



CPUs are mostly idle…

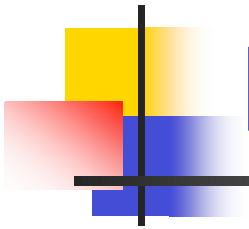
Solution + New Problem





Experiments and Approaches

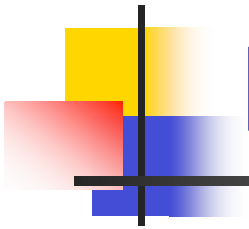
- Only Provide what Client Requests
  - Fast at low latency
  - Unacceptable Otherwise
- Return Requested Results Immediately & provide other results later
  - Separate server thread devoted to caching results
  - Current Solution
- Enhanced with OWL State Machine
  - Anticipates display of OWL expressions



Pitfalls of Precache Thread

Server Can't Keep Up

- Small tests show
  - Server generally only gets a few seconds behind
  - Some extreme cases gets two minutes behind (these are extreme cases)
  - Server is usually idle.
- Only need to scale to ten or so users.

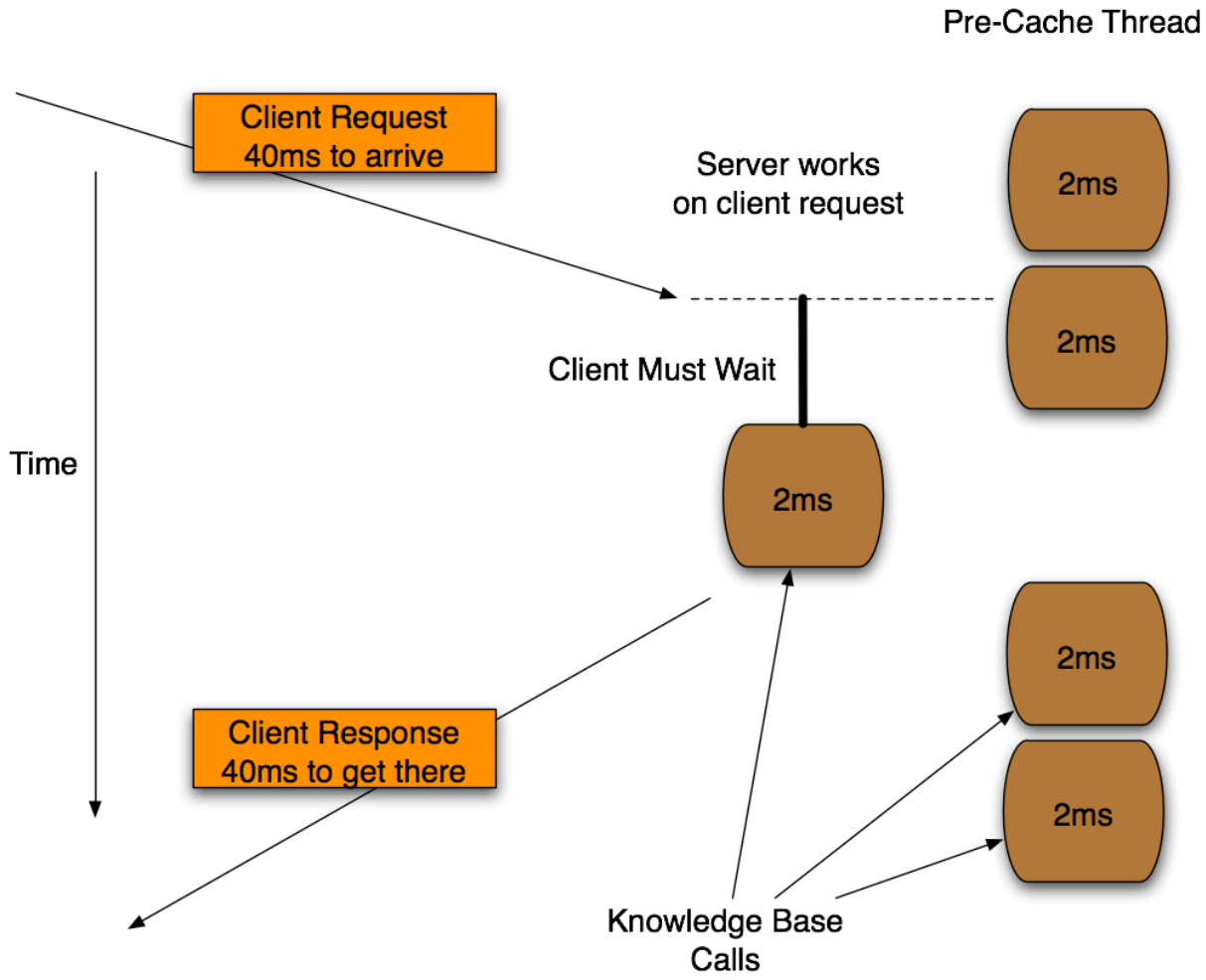


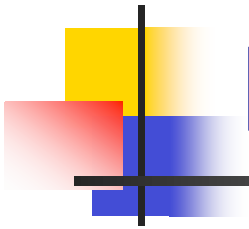
Pitfalls of Precache Thread

Server Response is Slower

- Issue is collisions requesting knowledge base lock
- Assume knowledge base calls take about 2ms
- On average, collision will occur when precache thread is halfway through its computation.
- Increase knowledge base calls to 3ms
- With network latency (80ms)
  - Change from 82ms to 83ms.

Precaching slows the server?



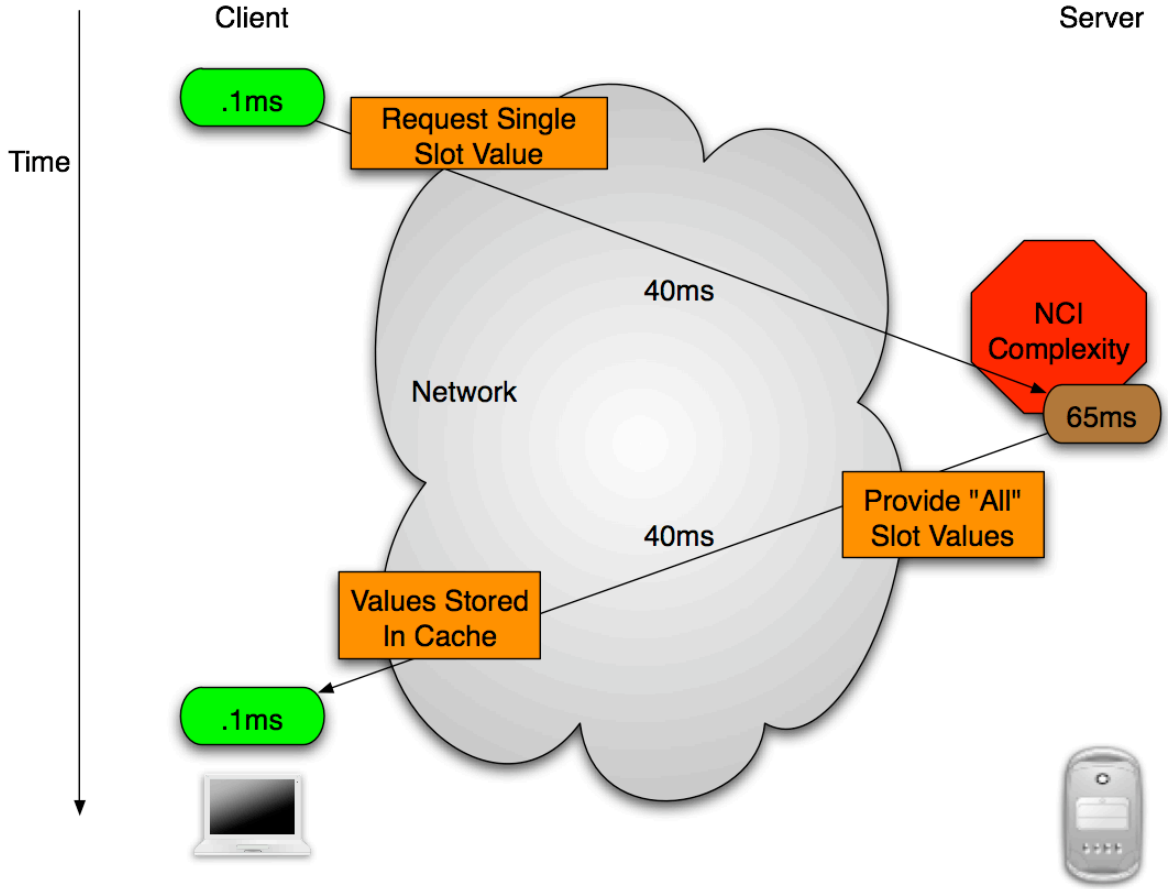


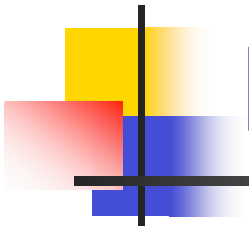
Pitfalls of Precache Thread

- Use of bandwidth
  - Generally does not use too much but
  - There are spikes
  - Short but at capacity of network
- Based on OS X Activity Monitor
- Impact Unknown
- Compressing Sockets?

Pitfalls of Precache Thread

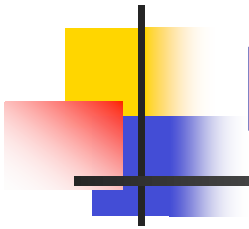
Deferred Results are Needed Now





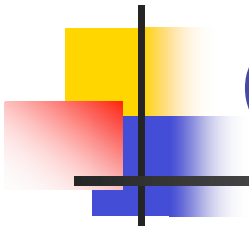
Role of Transactions

Begin Transaction
    do really tricky thing
Commit Transaction
Or Catch Failure and Rollback



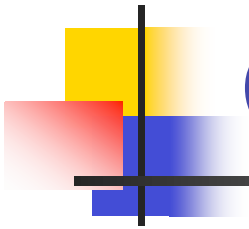
Role of Transactions

- Rollback on failure
- Protection from conflicting changes
  - Read Uncommitted
- Protection from premature update
  - Read Committed
- Protection from others updates
  - Repeatable Read
- Atomic Operations
  - Serializable



Caching and Transactions

- Read Uncommitted
  - One Cache
- Read Committed
  - My writes are not visible to others
- Repeatable Read
  - Start Transaction ⇒ Client Cache Emptied
- Serializable
  - Can’t anticipate clients needs during transaction



Caching and Transactions

- At Repeatable Read
  - Operation takes 40 seconds
  - CPU speed makes little difference
- At Read Committed
  - Operation takes 5 seconds
  - CPU speed is important
- Recommendation to NCI
  - Read Committed
- Cache hit rate is 99% (?!!)

OWL Overhead