

Analysis and Design of Large Scale Software II (SENG 401) Concept Mapping Application

<names>, <date>

SENG 401, <Group#>, Winter 2008

Table of Contents

| | |
|---|---|
| Executive Summary..... | 2 |
| 1 Conceptual View | 2 |
| 1.1 Global Analysis | 2 |
| 1.1.1 Factor Table..... | 2 |
| 1.1.2 Issue Cards..... | 2 |
| 1.1.2.1 Interface Speed and Memory Requirements | 2 |
| 1.1.2.2 Z-order | 3 |
| 1.2 Conceptual Configuration..... | 3 |
| 1.2.1 Application Level | 3 |
| 1.2.2 Supervisor Component | 4 |
| 1.3 Global Evaluation | 4 |
| 1.4 Resource Budgeting..... | 4 |
| 2 Module View..... | 5 |
| 2.1 Global Analysis | 5 |
| 2.1.1 Factor Table (additional and changed factors)..... | 5 |
| 2.1.2 Issue Cards (additional and changed issue cards)..... | 5 |
| 2.1.2.1 <title> | 5 |
| 2.2 Mapping Conceptual Elements to Modules..... | 5 |
| 2.3 Use Dependency Diagrams between Modules | 5 |
| 2.4 Diagrams Assigning Modules to Layers..... | 6 |
| 2.5 Use Dependency Diagrams between Layers..... | 7 |
| 2.6 Global Evaluation..... | 7 |
| 2.7 Interface Design..... | 7 |
| 2.7.1 Interface Definition: Security/Approval | 7 |
| 3 Execution View | 8 |
| 4 Code View | 8 |
| A1: Glossary..... | 8 |
| A2: Summary of design decisions | 8 |
| A3: <Any other relevant appendices...> | 8 |
| References | 8 |

Executive Summary

...

1 Conceptual View

...

1.1 Global Analysis

...

1.1.1 Factor Table

...

| Factor | Flexibility and Changeability | Impact |
|---|---|---|
| T1: Interface | | |
| T1.1: Speed of Interface | | |
| The interface must react to mouse events quickly. (Implicit requirement.) | There is little flexibility in this factor: while processes perceived by the user as complex may take time, mouse events should be next to instantaneous. | Components affected are the window pane and the selectable objects in the pane. |
| T1.2: System resources used | | |
| Windows has limited systems resources (memory, window handles, etc.), so the application must be conservative in its use of them. (Implicit requirement.) | If we were to move to an O/S with less overhead (UNIX?), then there might be looser constraints in the use of resources. | Components affected are the window pane, the selectable objects in the pane, and all graphical objects. |
| T1.3: Rectangular windows | | |
| Windows also assumes that all windows are rectangular in shape, while the application requires arbitrarily-shaped objects. | No flexibility | We will have to live with rectangular visual objects or "role our own". |
| P1: Concept map visualization | | |
| P1.1: Object Layers | | |
| Objects must be perceived by the user to be in "layers" – that is, users must be able to select overlapping objects that are "on top" and to control the relative "frontness" of overlapping objects. (Corollary of the "Easy Node Arrangement" requirement.) | This is not a flexible factor. | Components affected are the window pane, the selectable objects in the pane, and all graphical objects. |
| ... | | |

1.1.2 Issue Cards

...

| |
|---|
| 1.1.2.1 Interface Speed and Memory Requirements |
| All the objects in the concept mapping interface must react quickly to mouse events (selection, drag-and- |

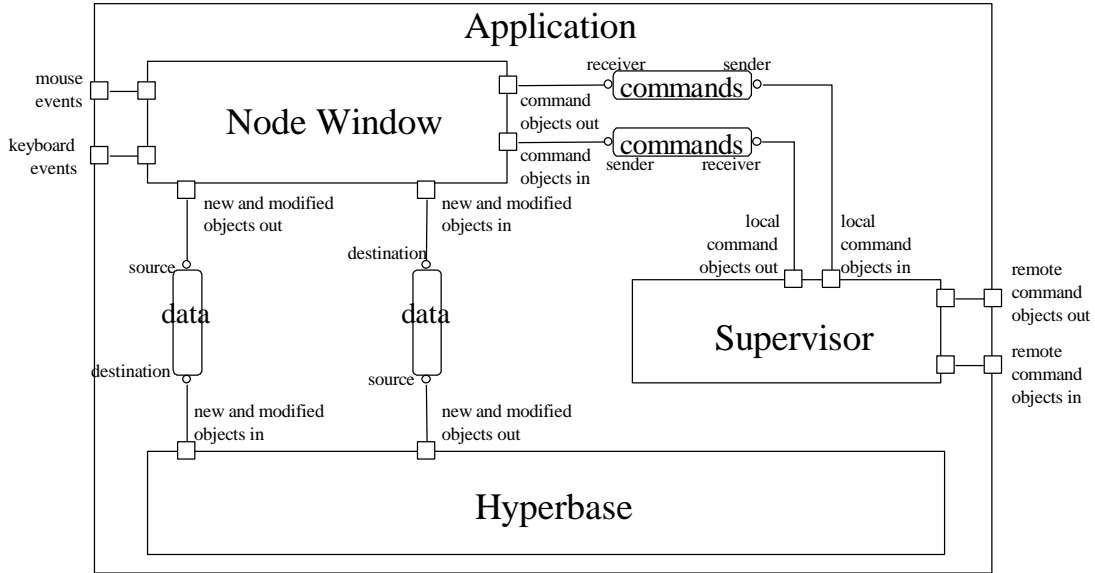
| |
|--|
| <p>drop, etc.), but must also take up limited memory.</p> <p>Influencing Factors:</p> <ul style="list-style-type: none"> • T1.1: Speed of Interface, T1.2: System resources used. While all the objects could easily be implemented as individual windows, windows constitute a huge overhead in terms of memory and resource usage. • T1.3: Rectangular windows. Windows also assumes that all windows are rectangular in shape, while the application requires arbitrarily-shaped objects, which would constitute an addition level of custom processing. |
| <p>Solution:</p> <p>The application will have to make use of a “roll your own” solution, where the background object will query the “contained” objects (in order) to determine if the clicked mouse location is contained within that object. (ref. Factors T1.1 and T1.2)</p> <p>Strategy: Variation on Chain of Responsibility (Gamma, Helm, Johnson & Vlissides, 1995)</p> <p>The background container object (node background) keeps an ordered list of objects. When the object receives a mouse event (and coordinates), it passes the coordinates to each object in turn until one accepts the responsibility to process the event (because the coordinates are within it’s borders). See also the “Z-order” issue card.</p> |
| <p>Related strategies:</p> <ul style="list-style-type: none"> • <i>Chain or Responsibility:</i> This is very similar to the Chain of Responsibility pattern, but in CoR, each object keeps a pointer to and calls the next object in the chain, whereas in this strategy, the container object has the responsibility of keeping a list of all pointer to contained objects and calling them each in turn. • <i>Z-order:</i> See the Z-order issue card. |

| |
|---|
| <p>1.1.2.2 Z-order</p> |
| <p>Each object has a location and clicking, collecting, dragging, drawing requires a consistent z-order. (ref. factor P1.1)</p> <p>Influencing Factors:</p> <ul style="list-style-type: none"> • ... |
| <p>Solution:</p> <p>Use a doubly linked list. When drawing start at the “bottom” and draw each in turn so top ones cover the bottom ones. When selecting, start at the top and work your way to the bottom.</p> <p>Strategy: Variation on Chain of Responsibility and Decorator (Gamma, Helm, Johnson & Vlissides, 1995)</p> <p>...</p> |
| <p>Related strategies:</p> <ul style="list-style-type: none"> • ... |

1.2 Conceptual Configuration

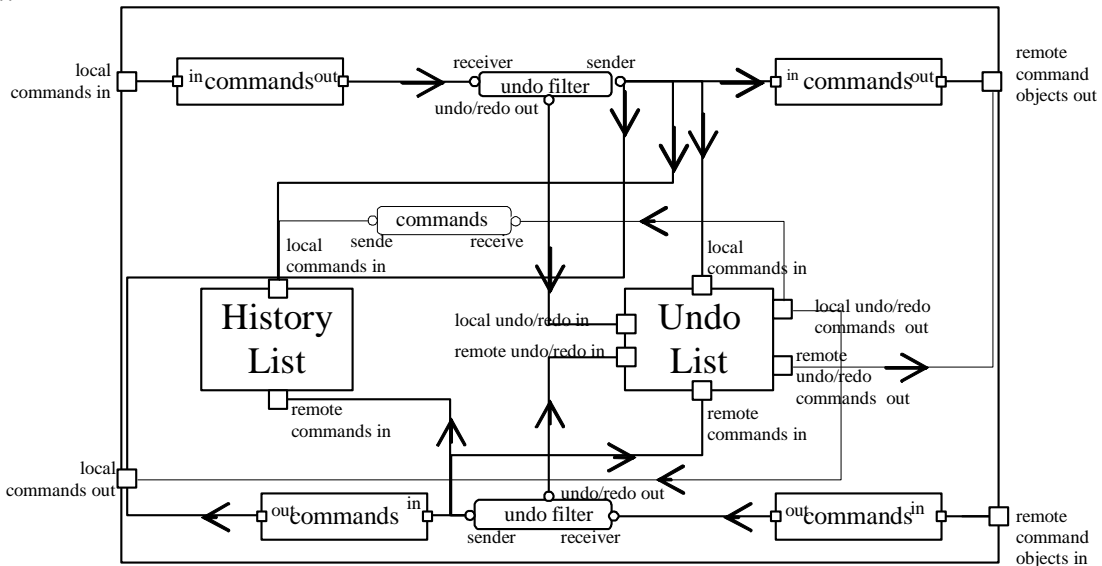
1.2.1 Application Level

<provide a prose overview and be sure to explain all the components and connectors referencing any detailed definitions that you have included>



1.2.2 Supervisor Component

...



1.3 Global Evaluation

...

1.4 Resource Budgeting

...

2 Module View

...

2.1 Global Analysis

...

2.1.1 Factor Table (additional and changed factors)

...

| Factor | Flexibility and Changeability | Impact |
|--------|-------------------------------|--------|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

2.1.2 Issue Cards (additional and changed issue cards)

...

| |
|---|
| 2.1.2.1 <title> |
| <p>...</p> <p>Influencing Factors:</p> <p>...</p> |
| <p>Solution:</p> <p>...</p> <p>Strategy:</p> <p>...</p> |
| <p>Related strategies:</p> <p>...</p> |

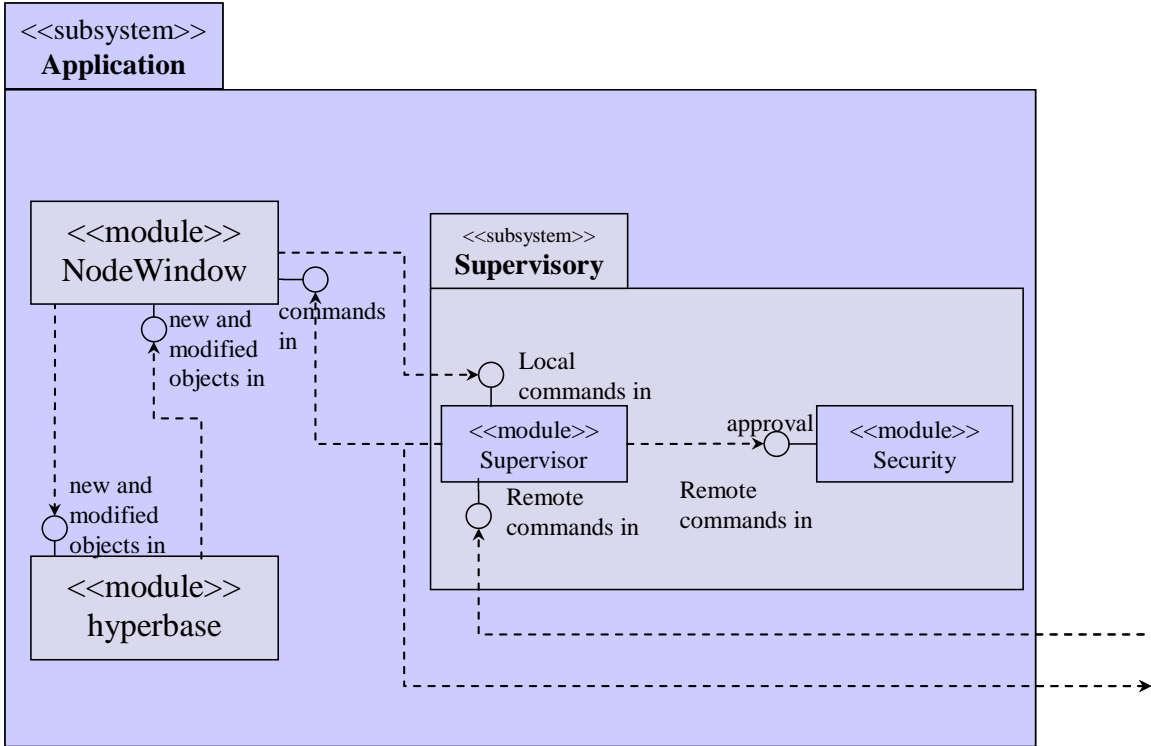
...

2.2 Mapping Conceptual Elements to Modules

...

2.3 Use Dependency Diagrams between Modules

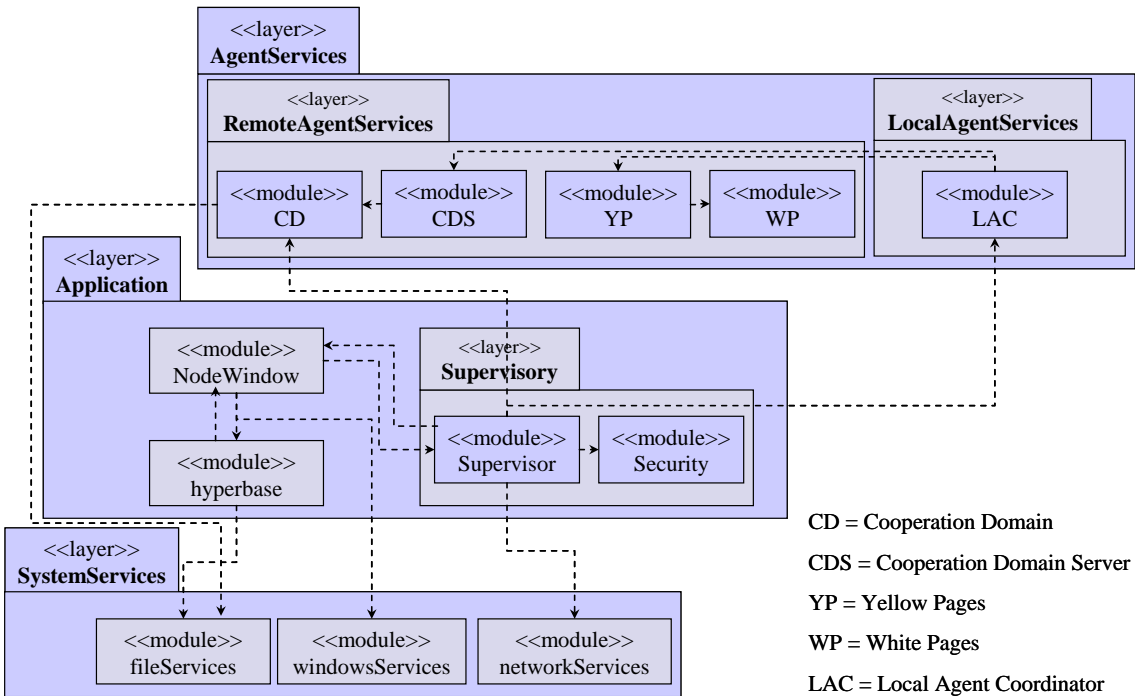
...



...

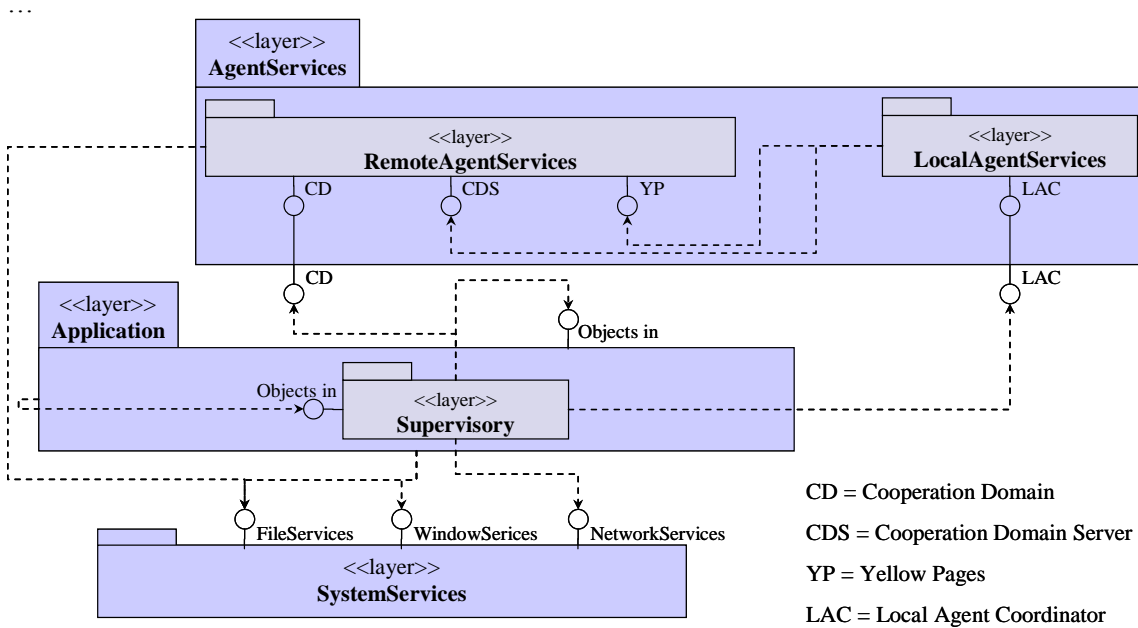
2.4 Diagrams Assigning Modules to Layers

...



...

2.5 Use Dependency Diagrams between Layers



2.6 Global Evaluation

...

2.7 Interface Design

...

2.7.1 Interface Definition: Security/Approval

RegisterObject(id, pin, object, securitySpec) → status

Registers securitySpec for object provided that id and pin allow it. Succeeds only if this is the first registration of object. id will become the owner of object.

ChangeObject(id, pin, object, securitySpec) → status

Updates security with securitySpec for object provided that id and pin allow it.

DeleteObject(id, pin, object) → status

Deletes object's security registration provided id and pin allow it

ApproveOp(id, pin, command) → status

Returns OK iff id and pin are allowed to execute command.

status: {OK, denied, unknownObject, invalidSecuritySpec, invalidPin, wrongPin, unknownID}

...

3 Execution View

<future assignment>

4 Code View

<future assignment>

A1: Glossary

...

A2: Summary of design decisions

...

A3: <Any other relevant appendices...>

...

References

(Gamma, Helm, Johnson & Vlissides, 1995) E. Gamma, R. Helm, R. Johnson, J. Vlissides (1995). Design Patterns. Addison Wesley. ISBN 0.201-63361-2.

...