|  |  |
| --- | --- |
| Patterns | Specification |
| Publish-subscribe | The system components need to communicate with each other  for example:  When new event occur such as new project/ homework to notify the student or when a change in the subject syllabus happens, the lecturer, head of department and the dean shall be notified.  List based publish subscriber: student who registered in a subject.  Content based: for department head master and the dean. |
| Model view controller | We want the user interface functionality be kept separate from application functionality and yet still be responsive to user input |
| SOA | We will use number of services that are offered by AAU service provider and our system need to communicate without any detailed knowledge of their implementation.  The tactic that will be used to insure the interoperability is tailored interface.  Asynchronous connectors will be used in conjunction  With the publish-subscribe pattern. |
| Layered pattern | We will use this pattern to achieve low coupling and high modifiability |
| Client server pattern | We will use this pattern because we want to manage a set of shared resource , plus we want to promote the modifiability in our project |
| Pipe filter | We will filter the data from the university database |
| Broker | We want to structure a distributed software and the client doesn’t need to know the nature and location of the service provided |

Main scenario - Sequence diagram

List of use cases – core use case with priority

Part of class diagram ll core use case

Tables for tactics and architecture patterns

Tracking of documents

* The milestone is base architecture