System Design Checklist

|  |  |
| --- | --- |
| **Reusability** | |
| Q1. What are the interfaces between the reused software and the operator? | The target system doesn’t reuse legacy software. The operators are designed for new generated classes in the target system. |
| Q2. What are the interfaces between the reused software and the plant system/environment? | The target system doesn’t interface between the plant system/environment and the reused software. |
| Q3. How does the architecture define reusable software for the target and legacy systems? | The architecture well defines external legacy systems but doesn’t define reusable component of the legacy systems. |
| Q4. What method does the design define to reuse the software? | The design doesn’t define reuse method but each class can be reused for the system extension. |
| Q5. What environment must be initialized prior to executing the reused software? | The target system uses Web-based environment but it doesn’t initialize Web-browser UI. |
| Q6. What schema does the project adopt to reuse software components in the design phase? | The project document doesn’t define a reusable software schema. |
| Q7. Does the each component in the design has specific, clear and well defined operations in the each interface? | The component diagram doesn’t define an interface for access. Each component designed roughly. |
| Q8. Is each component capable of handling repeated usage? | The components are not well designed for reuse but classes are well designed for repeated usage. |
| Analysis and Recommendation  The design document doesn’t consider software reusability for the legacy and the target system. To improve the system quality and to reduce project cost, it is recommended that the project adopts software reuse strategy and technology.  Especially, RAS(Reusable Asset Specification) of OMG is a set of guidelines and recommendations about the structure, content, and descriptions of reusable software assets. |  |

|  |  |
| --- | --- |
| **Efficiency** | |
| Q1. Does the potential component have run-time characteristics that are acceptable within the context of the existing design? |  |
| Q2. Does the potential component have data management capabilities that are consistent with the existing design? |  |
| Q3. Have algorithmic design alternatives been considered? If yes, why was this design chosen? |  |
| Q4. Has an object-relationship model been defined? | Yes; class diagram explains this |
| Q5. Is each subsystem appropriately allocated to processors and tasks? | The system uniformly runs its processes to carry out it’s tasks |
| Q6. How much memory is allocated to each component? | The amount of memory allocated isn’t explained in the plan |
| Q7. How is data communicated between software components? | The data flow diagram explains the process from which the user inputs information & commands, to the point where the system outputs its reponses accordingly |
| Q8. Can simpler data structures be used? | Yes, there is always room to simplify any form of data structures |
| Analysis and Recommendations |  |

|  |  |
| --- | --- |
| **Flexibility** | |
| Q1. Is the system capable of adapting to all types of problems or defects? |  |
| Q2. Is the system capable of allocating resources when under high stress? |  |
| Q3. Can the system retrieve information based on generalized queries/requests? | With each query there is a subclass, subtype to classify for |
| Q4. Can this system evolve? |  |
| Q5. Can we adjust for any changes in requirements, internal or external, made by the customer? |  |
| Q6. Can the system handle being introduced to new components? |  |
| Q7. Can the minimum and maximum information output be adjusted? |  |
| Q8. What types of risks is the system capable of handling? |  |
| Analysis and Recommendations |  |