Date of Assessment: Project: **Volunteers’ Management System**

Assessor(s):

Process Assessed:

|  |  | Y, N, NA | F, O | Comments |
| --- | --- | --- | --- | --- |
| General Confederations | | | | |
|  | The complexity of the system matches the functionality it provides. |  |  |  |
|  | The system has a single consistent, coherent architecture. |  |  |  |
|  | The number and types of component is reasonable. |  |  |  |
|  | The system has a consistent system-wide security facility. All the security components work together to safeguard the system. |  |  |  |
|  | The system will meet its availability targets. |  |  |  |
|  | The architecture will permit the system to be recovered in the event of a failure within the required amount of time. |  |  |  |
|  | The architecture provides defines clear interfaces to enable partitioning for parallel team development. |  |  |  |
|  | The designer of a model element can understand enough from the architecture to successfully design and develop the model element. |  |  |  |
|  | The proposed solution can be easily understood by someone generally knowledgeable in the problem domain. |  |  |  |
|  | All people on the team share the same view of the architecture as the one presented by the software architect. |  |  |  |
|  | The Software Architecture Document is current. |  |  |  |
|  | The Design Guidelines have been followed. |  |  |  |
|  | The key performance requirements (established budgets) have been satisfied. |  |  |  |
| **Architectural Analysis Considerations** | | | | |
|  | Subsystem and package partitioning and layering is logically consistent. |  |  |  |
|  | All analysis mechanisms have been identified and described. |  |  |  |
|  | **Sub-system related:** |  |  |  |
|  | The services (interfaces) of subsystems in upper-level layers have been defined. |  |  |  |
|  | The dependencies between subsystems and packages correspond to dependency relationships between the contained classes. |  |  |  |
|  | The classes in a subsystem support the services identified for the subsystem. |  |  |  |
|  | **Classes related:** |  |  |  |
|  | The key entity classes and their relationships have been identified. |  |  |  |
|  | Relationships between key entity classes have been defined. |  |  |  |
|  | The name and description of each class clearly reflects the role it plays. |  |  |  |
|  | The description of each class accurately captures the responsibilities of the class. |  |  |  |
|  | The entity classes have been mapped to analysis mechanisms where appropriate. |  |  |  |
|  | The role names of aggregations and associations accurately describe the relationship between the related classes. |  |  |  |
|  | The multiplicities of the relationships are correct. |  |  |  |
|  | The key entity classes and their relationships are consistent with the business model (if it exists), domain model (if it exists), requirements, and glossary entries |  |  |  |
| **General Model Considerations** | | | | |
|  | The model is at an appropriate level of detail given the model objectives. |  |  |  |
|  | For the business model, requirements model or the design model during the elaboration phase, there is not an over-emphasis on implementation issues. |  |  |  |
|  | For the design model in the construction phase, there is a good balance of functionality across the model elements, using composition of relatively simple elements to build a more complex design. |  |  |  |
|  | The model demonstrates familiarity and competence with the full breadth of modeling concepts applicable to the problem domain; modeling techniques are used appropriately for the problem at hand. |  |  |  |
|  | Concepts are modeled in the simplest way possible. |  |  |  |
|  | The model is easily evolved; expected changes can be easily accommodated. |  |  |  |
|  | At the same time, the model has not been overly structured to handle unlikely change, at the expense of simplicity and comprehensibility. |  |  |  |
|  | The key assumptions behind the model are documented and visible to reviewers of the model. If the assumptions are applicable to a given iteration, then the model should be able to be evolved within those assumptions, but not necessarily outside of those assumptions. Documenting assumptions is a way of indemnifying designers from not looking at "all" possible requirements. In an iterative process, it is impossible to analyze all possible requirements, and to define a model which will handle every future requirement. |  |  |  |
| Diagrams | | | | |
|  | The purpose of the diagram is clearly stated and easily understood. |  |  |  |
|  | The graphical layout is clean and clearly conveys the intended information. |  |  |  |
|  | The diagram conveys just enough to accomplish its objective, but no more. |  |  |  |
|  | Encapsulation is effectively used to hide detail and improve clarity. |  |  |  |
|  | Abstraction is effectively used to hide detail and improve clarity. |  |  |  |
|  | Placement of model elements effectively conveys relationships; similar or closely coupled elements are grouped together. |  |  |  |
|  | Relationships among model elements are easy to understand. |  |  |  |
|  | Labeling of model elements contributes to understanding. |  |  |  |
| **Documentation** | | | | |
|  | Each model element has a distinct purpose. |  |  |  |
|  | There are no superfluous model elements; each one plays an essential role in the system. |  |  |  |
| **Error recovery** | | | | |
|  | For each error or exception, a policy defines how the system is restored to a "normal" state. |  |  |  |
|  | For each possible type of input error from the user or wrong data from external systems, a policy defines how the system is restored to a "normal" state. |  |  |  |
|  | There is a consistently applied policy for handling exceptional situations. |  |  |  |
|  | There is a consistently applied policy for handling data corruption in the database. |  |  |  |
|  | There is a consistently applied policy for handling database unavailability, including whether data can still be entered into the system and stored later. |  |  |  |
|  | If data is exchanged between systems, there is a policy for how systems synchronize their views of the data. |  |  |  |
|  | In the system utilizes redundant processors or nodes to provide fault tolerance or high availability, there is a strategy for ensuring that no two processors or nodes can 'think' that they are primary, or that no processor or node is primary. |  |  |  |
|  | The failure modes for a distributed system have been identified and strategies defined for handling the failures. |  |  |  |
| **Transition and Installation** | | | | |
|  | The process for upgrading an existing system without loss of data or operational capability is defined and has been tested. |  |  |  |
|  | The process for converting data used by previous releases is defined and has been tested. |  |  |  |
|  | The amount of time and resources required to upgrade or install the product is well-understood and documented. |  |  |  |
|  | The functionality of the system can be activated one use case at a time. |  |  |  |
| **Administration** | | | | |
|  | Disk space can be reorganized or recovered while the system is running. |  |  |  |
|  | The responsibilities and procedures for system configuration have been identified and documented. |  |  |  |
|  | Access to the operating system or administration functions is restricted. |  |  |  |
|  | Licensing requirements are satisfied. |  |  |  |
|  | Diagnostics routines can be run while the system is running. |  |  |  |
|  | The system monitors operational performance itself (e.g. capacity threshold, critical performance threshold, resource exhaustion). |  |  |  |
|  | The policies and procedures for network (LAN, WAN) monitoring and administration are defined. |  |  |  |
|  | Faults on the network can be isolated. |  |  |  |
|  | There is an event tracing facility that can be enabled to aid in troubleshooting. |  |  |  |
|  | It is not possible for a malicious user to enter the system, or destroy critical data, or consume all resources. |  |  |  |
| **Performance** | | | | |
|  | Performance requirements are reasonable and reflect real constraints in the problem domain; their specification is not arbitrary. |  |  |  |
|  | Estimates of system performance exist (modeled as necessary using a Workload Analysis Model), and these indicate that the performance requirements are not significant risks. |  |  |  |
|  | System performance estimates have been validated using architectural prototypes, especially for performance-critical requirements. |  |  |  |
|  | Response time requirements for each message have been identified. |  |  |  |
| **Memory Utilization** | | | | |
|  | Memory budgets for the application have been defined. |  |  |  |
|  | Actions have been taken to detect and prevent memory leaks. |  |  |  |
|  | There is a consistently applied policy defining how the virtual memory system is used, monitored and tuned. |  |  |  |
| **Cost and Schedule** | | | | |
|  | The actual number of lines of code developed thus far agrees with the estimated lines of code at the current milestone. |  |  |  |
|  | The estimation assumptions have been reviewed and remain valid. |  |  |  |
|  | Cost and schedule estimates have been re-computed using the most recent actual project experience and productivity performance. |  |  |  |
| **Portability** | | | | |
|  | Portability requirements have been met. |  |  |  |
|  | Programming Guidelines provide specific guidance on creating portable code. |  |  |  |
|  | Design Guidelines provide specific guidance on designing portable applications. |  |  |  |
| **Reliability** | | | | |
|  | Measures of quality (MTBF, number of outstanding defects, etc.) have been met. |  |  |  |
|  | The architecture provides for recovery in the event of disaster or system failure |  |  |  |
| **Security** | | | | |
|  | Security requirements have been met. |  |  |  |
| **The Logical View** | | | | |
|  | Accurately and completely presents an overview of the architecturally significant elements of the design. |  |  |  |
|  | Presents the complete set of architectural mechanisms used in the design along with the rationale used in their selection. |  |  |  |
|  | Presents the layering of the design, along with the rationale used to partition the layers. |  |  |  |
| **The Process View** | | | | |
|  | Potential race conditions (process competition for critical resources) have been identified and avoidance and resolution strategies have been defined. |  |  |  |
|  | The system tolerant of errors and exceptions, such that when an error or exception occurs, the system can revert to a consistent state. |  |  |  |
|  | Processes are sufficiently independent of one another that they can be distributed across processors or nodes when required. |  |  |  |
| **The Deployment View** | | | | |
|  | The throughput requirements have been satisfied by the distribution of processing across nodes, and potential performance bottlenecks have been addressed. |  |  |  |
|  | Requirements for reliable transport of messages, such that they exist, have been satisfied. |  |  |  |
|  | Requirements for secure transport of messages, such that they exist, have been satisfied. |  |  |  |
|  | System availability requirements, to the extent that they exist, have been satisfied. |  |  |  |
|  | All potential failure modes have been documented. |  |  |  |
|  | Faults in the network can be isolated, diagnosed and resolved. |  |  |  |
|  | There is a stated policy for the actions to be taken when the maximum CPU utilization is exceeded |  |  |  |
|  | | | | |
|  | | | | |
|  | | | | |
| *REFERENCE ITEMS/DOCUMENTS* | | | | |
| *RUP Documentation* | | | | |

# COMMENTS PAGE

|  |  |
| --- | --- |
| **#** | **Comments from assessment** |
| **1.** |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |