

Design Review Checklist

22/03/2017

0.1

|  |  |
| --- | --- |
| **Nombre.** | **Matricula** |
| **Jesús Alberto Goiz Barrales.** | **24400085** |
| **Juan Alberto Gutiérrez Canto.** | **24400063** |
| **Brenda Robles Antonio.** | **24400073** |
| **Rene Moratilla Montes.** | **24400075** |
| **Guillermo Vivaldo Vazquez** | **24400093** |

**Control De Versiones**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre Del Archivo | Versión | Fecha | Autor | Comentarios |
| Csoft-Diseño\_Functional-Specification\_Template-200317.docx | 0.1 | 20/03/17 | GVV | Creación Del Documento |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Table C65 C++ PSP2.1 Design Review Checklist**

PROGRAM NAME AND #: Alta De Planes

1. Template de Especificación Funcional
2. Template de Escenario Operacional
3. Template de Especificación de Estados
4. Templates de Especificación Lógica

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Purpose | To guide you in conducting an effective design review | 1 | 2 | 3 | 4 |
| General | • As you complete each review step, check off that item in the box to the right.  • Complete the checklist for one program unit before you start to review the next unit. | \* | \* | \* | \* |
| Complete | Ensure the requirements, specifications, and high-level design are completely covered by the design:  • All specified outputs are produced.  • All needed inputs are furnished.  • All required includes are stated. | \* | \* | \* | \* |
| State Machine | Verify the state machine design:  • The structure has no hidden traps or loops.  • It is complete; that is, all possible states have been identified.  • It is orthogonal; that is, for every set of conditions there is one and only one possible next state.  • The transitions from each state are complete and orthogonal. That is, from every state, a unique next state is defined for every possible combination of state machine input values. | ***\**** | ***\**** | ***\**** | ***\**** |
| Logic | • Verify that program sequencing is proper:   * Stacks, lists, and so on are in the proper order. * Recursion unwinds properly.   • Verify that all loops are properly initiated, incremented, and terminated. | \* | \* | \* | \* |
| Special Cases | Check all special cases:  • Ensure proper operation with empty, full, minimum, maximum, negative, zero values for all variables.  • Protect against out-of-limits, overflow, underflow conditions.  • Ensure “impossible” conditions are absolutely impossible.  • Handles all incorrect input conditions. | \* | \* | \* | \* |
| Functional Use | • Verify that all functions, procedures, or objects are fully understood and properly used.  • Verify that all externally referenced abstractions are precisely defined. | \* | \* | \* | \* |
| Names | Verify the following.  • All special names and types are clear or specifically defined.  • The scopes of all variables and parameters are self-evident or defined.  • All named objects are used within their declared scopes. | \* | \* | \* | \* |
| Standards | Review the design for conformance to all applicable design standards. | \* | \* | \* | \* |