# Rúbricas para la evaluación de la materia

Gunnar Eyal Wolf Iszaevich

Sistemas Operativos; Facultad de Ingeniería, UNAM

## Índice



## 1. ¿Rúbricas?

Mi intención como docente es facilitarles a ustedes (alumnos) herramientas para que puedan estimar la calificación de los proyectos a desarrollar que tendrán en el transcurso del semestre, haciendo tan claros, objetivos y expresos los criterios que emplearé para evaluarlos.

En este documento les presento las *rúbricas* (también conocidas como *matrices de evaluación*) que emplearé a lo largo del semestre.

Cabe mencionar que, como se los comenté en las primeras clases, estoy experimentando con esta forma de trabajo, por lo que es posible que requiera ajustar los criterios. Me comprometo a hacer oportunamente de su conocimiento cualquier cambio que les haga, y los invito a indicarme si creen que cualquiera de los aspectos parece excesivo o podría mejorarse — ¡No por ser el docente tengo siempre la razón! ;-)

# 2. Exposición frente al grupo

La evaluación de la exposición considera factores tanto de la calidad del material desarrollado como de la forma en que se presenta al grupo, como puede apreciarse en la siguiente tabla.

Toda exposición se evaluará considerando como requisitos indispensables:

- El material presentado debe ser desarrollado íntegramente por el o los alumnos expositores (con citas textuales indicadas expresamente) o será automáticamente descalificado.
- La elección de tema y fecha deben ser acordadas previamente con el profesor, con no menos de dos semanas de anticipación.

La exposición ante el grupo constará de dos calificaciones: Un 70 % asignado por el profesor, y un 30 % proveniente de evaluación grupal en que los compañeros presentes en la sesión evalúen cuantitativamente y de forma anónima.

## 2.1. Criterios de evaluación grupal

- Originalidad
- Relevancia
- Claridad en la presentación
- Presencia

Los compañeros tendrán también un campo para hacer comentarios en formato libre al ponente.

## 2.2. Criterios de evaluación por el docente

| | **Excelente** (100 %) | **Bueno** (75 %) | **Suficiente** (50 %) | **Insuficiente** (0 %) | Peso |
|---|---|---|---|---|---|
| **Originalidad del tema** | Tema novedoso propuesto a iniciativa del expositor | Tema complejo elegido directamente de los temas ejemplo | Tema sencillo elegido directamente de los temas ejemplo | | 10 % |
| **Material desarrollado** | Presentación acompañada del material desarrollado en formato de reporte/artículo, 4-8 páginas, enviado para comentarios con 3 o más días de antelación | Presentación acompañada del material desarrollado en prosa (distinta longitud), o enviado para comentarios con muy poca antelación | Únicamente presentación, o no enviado para comentarios con anticipación | No se entregó material | 20 % |

Continued on next page

Continued from previous page

| | **Excelente** (100 %) | **Bueno** (75 %) | **Suficiente** (50 %) | **Insuficiente** (0 %) | Peso |
|---|---|---|---|---|---|
| **Contenido** | Cubre todos los puntos relevantes del tema abordado de forma clara y organizada lógicamente | Cubre mayormente el tema abordado manteniendo una organización lógica | Logra una cobertura parcial del tema o su organización entorpece la comprensión | La información presentada está incompleta o carece de un hilo conducente | 20 % |
| **Fuentes bibliográficas** | Se refiere a publicaciones especializadas, artículos de investigación, estado del arte en el campo | Cita recursos formales de consulta | Cita únicamente recursos no formales | No menciona referencias | 10 % |
| **Uso del tiempo** | Exposición en 15-20 minutos, buen tiempo para preguntas y respuestas | Exposición en 10-15 o en 20-25 minutos | Exposición menor a 15 minutos o mayor a 25 minutos (¡el profesor puede haberla interrumpido!) | | 10 % |
| **Dominio del tema** | Amplio conocimiento del tema incluso más allá del material expuesto; presenta con claridad y responde las preguntas pertinentes de los compañeros | Buen conocimiento del tema; presenta con fluidez, pero permanece claramente dentro del material presentado | Conocimiento suficiente del tema para presentarlo siguiendo necesariamente el material; responde sólo las preguntas más simples | No demuestra haber comprendido la información, depende por completo de la lectura del material para presentar, y no puede responder preguntas sencillas | 15 % |
| **Presencia** | Buen contacto ocular mantenido a lo largo de la sesión, presentación fluida, voz clara y segura | Buen contacto ocular, tal vez frecuentemente interrumpido por referirse a las notas. Presentación ligeramente carente de fluidez/seguridad | Contacto ocular ocasional por mantenerse leyendo la presentación. Voz baja o insegura. | Sin contacto ocular por leer prácticamente la totalidad del material. El ponente murmulla, se atora con la pronunciación de términos, cuesta seguirlo | 15 % |

# 3. Proyectos de desarrollo

A lo largo del semestre se entregarán varios proyectos de desarrollo de software, para evaluar las principales unidades del curso. La naturaleza de cada uno de estos proyectos será explicada en clase no menos de dos semanas antes de la fecha de entrega, y quedará disponible en el sitio de la materia y en el depósito Git.

El criterio para la calificación de estos proyectos se presenta a continuación; dentro de cada categoría (*Proyecto*, *Entrega*, *Legibilidad*, *Documentación*), el valor de cada uno de sus incisos será el mismo. Sólo se aceptará una entrega por persona o equipo.

## 3.1. Monitor de sistema

El primer proyecto es programar un monitor de recursos del sistema operativo (de los recursos que juzguen útiles, interesantes, o que decidan por la razón que sea), pero con la particularidad o requisito de que *debe trabajar* de forma concurrente, sincronizando multihilos, multiprocesos o ambos. Empleen los mecanismos de sincronización que consideren adecuados.

| | **Excelente** (100 %) | **Bueno** (75 %) | **Suficiente** (50 %) | **Insuficiente** (0 %) | Peso |
|---|---|---|---|---|---|
| **Requisitos** | | | | | 20 % |
| Cumplimiento | Se cumplen los requisitos planteados al 100 % e incluso se exceden | Los requisitos planteados se cumplen a cabalidad | El proyecto se aproxima a los requisitos, sin llegar a cumplirlos por completo | El proyecto presentado no tiene relación con lo solicitado en clase | |
| **Proyecto** | | | | | 20 % |
| Creatividad | El problema abordado es novedoso, o la manera en que aborda la necesidad es notoriamente original | Resuelve una necesidad ampliamente conocida, pero de forma original/interesante | Emplea mecanismos y fuentes de datos conocidas para resolver un problema ampliamente conocido | | |
| Complejidad | Combina datos de varias fuentes para brindar un panorama no obvio. | | Presenta directamente la información obtenida de una fuente sin aplicar procesamiento alguno. | | |
| Interfaz usuario | Logra una interfaz usuario atractiva, consistente y clara, con la explicación necesaria para un uso natural y fluido | La interfaz usuario muestra trabajo, pero requiere consultar la documentación para comprender el uso. | La interfaz usuario es suficiente para presentar los datos generados, pero su uso requiere comprender el código fuente. | El programa es imposible de utilizar exitosamente sin conocer la implementación detalladamente | |
| **Documentación** | | | | | 20 % |
| Documentación externa | Incluye una sección de presentación con, por lo menos: Nombres de los participantes del equipo, problema que busca resolver, lógica de operación (cómo lo resuelve), y ejemplos de uso o invocación | Incluye tres de los puntos mencionados | Incluye dos de los puntos mencionados | No incluye documentación del proyecto. | |

Continued on next page

Continued from previous page

| | **Excelente** (100 %) | **Bueno** (75 %) | **Suficiente** (50 %) | **Insuficiente** (0 %) | Peso |
|---|---|---|---|---|---|
| Entorno y dependencias | Detalla el entorno para el cual el programa fue escrito, detallando según sea pertinente lenguaje (incluyendo la implementación y versiones mayores), principales módulos que deben ser instalados (con sus respectivas versiones), y demás instrucciones pertinentes | | Indica los principales componentes requeridos para la construcción y ejecución del proyecto, pero omite detalles importantes que dificultan su exitosa ejecución. | | |
| Comentarios | El código está comentado donde hace falta, no repite información obvia. Los comentarios ayudan a comprender la lógica, no la implementación. | El código está comentado donde hace falta, pero los comentarios son excesivos: Además de la lógica general, mencionan lo obvio. | Hay algunos comentarios útiles en el programa, pero falta mucho para que ayude a una buena comprensión. | No hay comentarios. | |
| **Entrega** | | | | | 20 % |
| Historia en Git | El proyecto consta de un mínimo de cinco commits, con información suficientemente descriptiva para comprender el proceso de desarrollo | Consta de un mínimo de cinco *commits*, pero no presentan un título/comentario suficiente para entender el proceso de desarrollo | La entrega consta de un sólo *commit*, no permite entender el proceso de desarrollo del proyecto | No entregó usando Git | |
| Directorio de proyecto | El trabajo entregado consta exclusivamente del código fuente y la documentación, en una estructura acorde para su construcción/compilación, evaluación y uso directo; en caso de requerirlo, un `.gitignore` mantiene limpio el directorio al compilar. | Sólo el código fuente y la documentación forman parte de los *commits*, pero construir / ejecutar el código *ensucia* al repositorio (¿debería manejar `.gitignore`?) | El trabajo entregado incluye archivos innecesarios (como archivos objeto ya compilados o subdirectorios generados por el entorno de desarrollo empleado) | No entregó usando Git | |
| Código válido | Al ejecutar las instrucciones documentadas, el código puede ejecutarse exitosamente al primer intento. | Las instrucciones que forman parte de la documentación tienen que adecuarse para poder ejecutar el código, o hay errores menores que corregir para que funcione. | No está documentado cómo ejecutar el código, o hay errores mayores que corregir para poder ejecutarlo. | Resultó imposible probar la ejecución. | |
| **Concurrencia** | | | | | 20 % |

Continued on next page

Continued from previous page

| | **Excelente** (100 %) | **Bueno** (75 %) | **Suficiente** (50 %) | **Insuficiente** (0 %) | Peso |
|---|---|---|---|---|---|
| Multiproceso | Los diferentes hilos o procesos siguen una división clara de funciones o responsabilidades, o abonan en conjunto de forma determinante mejor de lo que lo harían en secuencial. La ejecución concurrente se mantiene durante la mayor parte del tiempo de ejecución. El multiprocesamiento es claramente beneficioso para la resolución del problema. | Hay manejo de múltiples hilos o procesos en momentos específicos; vemos más un intercambio de control que un avance en paralelo. | | Un sólo hilo / proceso, no implementa paralelismo | |
| Sincronización | Emplea mecanismos de sincronización en diferentes lugares del código, y siguiend diferentes tareas, o empleó mecanismos de sincronización distintos de los estudiados en clase (mutex, semáforo). Esto es, se nota un espíritu creativo en el uso de estos mecanismos. | Emplea mutex o semáforos para la sincronización, siguiendo los patrones estudiados. Encontramos el uso de por lo menos dos patrones diferentes. | Únicamente emplea mutex para proteger secciones críticas | No emplea mecanismos de sincronización | |

## 3.2. (Micro) sistema de archivos

Un (micro) sistema de archivos. Un programa que tome a un archivo y, dentro de él, permita operaciones básicas como si fuera un sistema de archivos. Las operaciones mínimas que debe proveer son:

- Listar el directorio
- Crear o eliminar un archivo
- Manejo básico de archivos en modos de lectura, escritura, y para agregar

El programa debe permitir un uso interactivo (como si fuera un shell del sistema operativo). Pueden desarrollarlo en el lenguaje que prefieran, pero el resultado debe ser funcional. Pueden incluir el sistema de archivos o indicarme cómo generarlo. Cuenta *perfectamente* como uso interactivo el ofrecerle una interfaz al sistema operativo para que cualquier programa pueda utilizarse dentro del (micro)sistema, por ejemplo, desarrollando un módulo de FUSE.

Basta con que representen un *directorio plano*, con soporte para un mínimo de cuatro entradas (archivos) diferentes. Son libres de elegir otros límites (por ejemplo, longitud de nombre de archivo, tamaño máximo de cada archivo, etc.) pero les pido que documenten *todos* los límites que puedan identificar.

| | **Excelente** (100 %) | **Bueno** (75 %) | **Suficiente** (50 %) | **Insuficiente** (0 %) | Peso |
|---|---|---|---|---|---|
| **Requisitos** | | | | | 20 % |
| Cumplimiento | Se cumplen los requisitos planteados al 100 % e incluso se exceden | Los requisitos planteados se cumplen a cabalidad | El proyecto se aproxima a los requisitos, sin llegar a cumplirlos por completo | El proyecto presentado no tiene suficiente relación con lo solicitado en clase | |
| **Proyecto** | | | | | 20 % |
| Complejidad | El mecanismo de representación o la manera de organizarlo son novedosos, sorprendentes | La representación de los datos resulta interesante, no obvia | Los datos se representan empleando las estructuras más naturales para tal fin | | |
| Claridad de desarrollo | El código, acompañado de la documentación, resultan de clara comprensión, a pesar de cualquier nivel de complejidad que mantengan | El código es ofuscado, dificil de seguir, pero la lectura de la documentación ayuda a hacerlo | Es dificil comprender la forma en que se desarrolló incluso teniendo la documentación, o esta está incompleta y no cubre este aspecto | Imposible de comprender | |
| Interfaz usuario | Logra una interfaz usuario atractiva, consistente y clara, con la explicación necesaria para un uso natural y fluido. Nota: Un módulo de núcleo o de FUSE, que pueda ser tranparentemente empleado desde terceros programas, cae aquí. | La interfaz usuario muestra trabajo, pero requiere consultar la documentación para comprender el uso. | La interfaz usuario es suficiente para presentar y manipular los datos, pero su uso requiere comprender el código fuente. | El programa es imposible de utilizar exitosamente sin conocer la implementación detalladamente | |
| **Documentación** | | | | | 20 % |
| Documentación externa | Incluye una sección de presentación con, por lo menos: Nombres de los participantes del equipo, problema que busca resolver, lógica de operación (cómo lo resuelve), y ejemplos de uso o invocación | Incluye tres de los puntos mencionados | Incluye dos de los puntos mencionados | No incluye documentación del proyecto. | |
| Entorno y dependencias | Presenta el entorno para el cual el programa fue escrito, detallando según sea pertinente lenguaje (incluyendo la implementación y versión mayor), principales módulos que deben ser instalados (con sus respectivas versiones), y demás instrucciones pertinentes | | Indica los principales componentes requeridos para la construcción y ejecución del proyecto, pero omite detalles importantes que dificultan su exitosa ejecución. | | |

Continued on next page

Continued from previous page

| | **Excelente** (100 %) | **Bueno** (75 %) | **Suficiente** (50 %) | **Insuficiente** (0 %) | Peso |
|---|---|---|---|---|---|
| Comentarios | El código está comentado donde hace falta, no repite información obvia. Los comentarios ayudan a comprender la lógica, no la implementación. | El código está comentado donde hace falta, pero los comentarios son excesivos: Además de la lógica general, mencionan lo obvio. | Hay algunos comentarios útiles en el programa, pero falta mucho para que ayude a una buena comprensión. | No hay comentarios. | |
| **Entrega** | | | | | 20 % |
| Historia en Git | El proyecto consta de un mínimo de cinco commits, con información suficientemente descriptiva para comprender el proceso de desarrollo | Consta de un mínimo de cinco *commits*, pero no presentan un título/comentario suficiente para entender el proceso de desarrollo | La entrega consta de un sólo *commit*, no permite entender el proceso de desarrollo del proyecto | No entregó usando Git | |
| Directorio de proyecto | El trabajo entregado consta exclusivamente del código fuente y la documentación, en una estructura acorde para su construcción/compilación, evaluación y uso directo; en caso de requerirlo, un `.gitignore` mantiene limpio el directorio al compilar. | Sólo el código fuente y la documentación forman parte de los *commits*, pero construir / ejecutar el código *ensucia* al repositorio (¿debería manejar `.gitignore`?) | El trabajo entregado incluye archivos innecesarios (como archivos objeto ya compilados o subdirectorios generados por el entorno de desarrollo empleado) | No entregó usando Git | |
| Código válido | Al ejecutar las instrucciones documentadas, el código puede ejecutarse exitosamente al primer intento. | Las instrucciones que forman parte de la documentación tienen que adecuarse para poder ejecutar el código, o hay errores menores que corregir para que funcione. | No está documentado cómo ejecutar el código, o hay errores mayores que corregir para poder ejecutarlo. | Resultó imposible probar la ejecución. | |
| **Legibilidad** | | | | | 20 % |
| Estructura | El código está bien organizado y emplea un estilo de indentación de forma consistente. | El código está mayormente indentado; hay inconsistencias menores. | Falta claridad en los bloques por no emplear indentación o hacerlo de forma absolutamente inconsistente. | | |
| Nomenclatura | Los nombres de los símbolos (variables, funciones, métodos, clases) son claros y acorde a su función; los principales elementos están documentados expresamente. | Los nombres de los símbolos mencionados son claros y acorde a su función, aunque no estén documentados. | Los nombres de los símbolos no son claros, pero su uso y significado forma parte de la documentación. | Cuesta trabajo seguir la lógica; los símbolos empleados no tienen nombres significativos, y su función no está documentada. | |