Ingeniería de Software

Construcción
Clase 1

Swebok: Guide to the software engineering body of knowledge
V3 2014 – Chapter 3 Software Construction

Sebastián Pizard

Universidad de la República

Agenda

- Estándares de Codificación
- Reutilización de Código
- Documentación del Código
- TDD
- Refactorización
- Programación de a Pares

Construcción de Software

- Refiere a la creación detallada del software mediante una combinación de codificación, verificación, pruebas unitarias, pruebas de integración y debugging.

Fundamentos de la Construcción de Software

- Minimización de la complejidad.
  - El software tiene que ser entendido por personas.
  - Reducir la complejidad es crítico para las pruebas.
  - Se puede lograr mediante: escribir código simple y legible en lugar de inteligente, el uso de estándares, diseño modular.
- Anticipación del cambio.

Fundamentos de la Construcción de Software

- Construcción para la verificación.
  - Permitir encontrar las faltas facilmente en el sw.
- Reuso.
  - Distintos niveles: abstracción / objeto / componente / sistema.
  - Contruir para el reuso / construir con reuso.
- Estandares en la construcción.
  - Externos / Internos.

Gestión de la Construcción

- Lo que consideramos “construcción” depende del modelo del ciclo de vida utilizado.
- Implica la elección del método de construcción. El orden en el cual se crean los componentes, se integran, se gestiona la calidad de la construcción y la asignación a los distintos programadores.

Consideraciones Prácticas

- El diseño en la construcción.
- Lenguajes de construcción
  - Lenguajes de configuración
  - Lenguajes caja de herramientas (toolkit)
  - Lenguajes de programación
  - Notaciones linguisticas
  - Notaciones formales
  - Notaciones visuales

Consideraciones Prácticas

- Codificación
  - Técnicas para crear código fuente, convenciones de código y plantillas de código fuente.
  - Uso de clases, tipos enumerados, variables, etc.
  - Uso de estructuras de control.
  - Manejo de excepciones (errores).
  - Prevención de rupturas de seguridad a nivel del código (por ej. índice fuera del array).
  - Mecanismos de exclusión y manejo de recursos compartidos.
  - Organización del código.
  - Documentación del código.
  - Puesta a punto del código (tuning).

Consideraciones Prácticas

- Pruebas en la construcción
  - Pruebas unitarias.
  - Pruebas de integración.

  El propósito es reducir el intervalo entre que se introduce una falla en el código y su detección.
- Construcción para el reuso
  - Análisis de variabilidad y diseño.
- Construcción con reuso
  - Involucra construir con el reuso de software existente.

Consideraciones Prácticas

- Calidad en la construcción
  - Pruebas unitarias y de integración.
  - Desarrollo basado en pruebas.
  - Uso de aserciones y programación defensiva.
  - Debugging.
  - Inspecciones.
  - Revisiones técnicas.
  - Análisis estático.
- Integración
  - Big bang
  - Incremental

Tecnologías de Construcción

- APIs
  - Application programming interface
  - Corresponde a un conjunto de firmas que son exportadas y disponibles para que los usuarios de un biblioteca o framework puedan escribir sus aplicaciones.
- Cuestiones de ejecución OO
  - Polimorfismo — es la habilidad de un lenguaje de soportar operaciones generales sin saber hasta tiempo de ejecución que tipos concretos de objetos incluirán.
  - Reflection — es la habilidad de un programa de observar y modificar su propia estructura y comportamiento en tiempo de ejecución.

Tecnologías de Construcción

- Tipos genéricos
  - Permiten definir un tipo o clase sin especificar los tipos de los objetos que utiliza.
- Aserciones, Diseño por contrato y Programación defensiva
  - Aserciones — es un predicado ejecutable que permite realizar chequeos en tiempo de ejecución. En general son quitadas para la versión del software que será puesta en producción.
  - Diseño por contrato — pre y poscondiciones son incluidas para cada operación.
  - Programación defensiva — la idea es proteger a las rutinas de entradas incorrectas.

Tecnologías de Construcción

- Manejo de errores, Manejo de excepciones y Tolerancia a las Fallas
  - La manera en que se manejan los errores afecta varios atributos de calidad (correctitud, robustez, etc). Existen muchas técnicas para realizar esto: aserciones, retornar valores neutros, logueo de errores, retornar códigos de error, “apagar el software”, etc.
  - Excepciones — se utilizan para detectar y procesar errores o eventos exepcionales. La estructura básica utiliza: throw para lanzar una excepción y el bloque try-catch para su manejo.
  - Tolerancia a las fallas — son un conjunto de técnicas que buscan aumentar la confiabilidad en el software mediante la detección de error y la recuperación.

Tecnologías de Construcción

- Modelos ejecutables
  - Buscan abstraer los detalles de los lenguajes específicos de programación y las decisiones sobre la organización del software.
  - Una especificación construida en un modelo ejecutable puede ser desplegada en varios entornos de ejecución sin realizar cambios.
  - Ejemplos son xUML, BPMN, Model-driven Architecture.

Tecnologías de Construcción

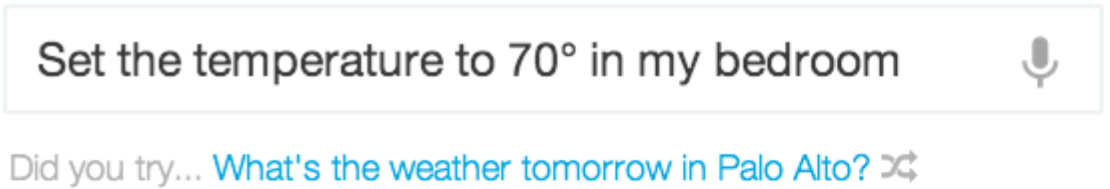
- Técnicas de construcción basadas en estados o dirigidas por tablas
  - Programación basada en estados — es una tecnología de programación que utiliza máquinas de estados finitos para describir el comportamiento de los programas.
  - Un método dirigido por tabla es un esquema en el cual se busca en una tabla información en lugar de utilizar sentencias lógicas (por ejemplo if-else).

Tecnologías de Construcción

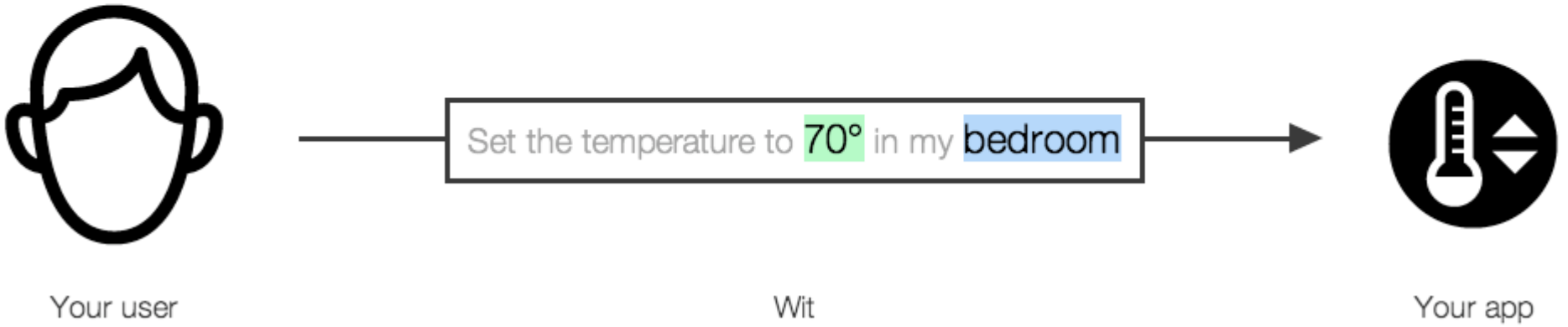
- Configuración en tiempo de ejecución e Internacionalización
  - Configuración en tiempo de ejecución (runtime configuration) — es una técnica para enlazar valores de las variables y seteos del programa en ejecución, utilizando, en general, archivos de configuración.
  - Internacionalización — es la actividad técnica de preparar un programa para multiples sitios. La actividad de localización corresponde a la adaptación para un lenguaje local específico.

Tecnologías de Construcción

- Procesamiento de entradas basado en la gramática
  - Involucra el análisis sintático (o parseo) de la entrada.




```
intent      = heating control
temperature = 70°F
where       = master_bedroom
```



Tecnologías de Construcción

- Primitivas de Concurrencia
  - Son primitivas que facilitan concurrencia y sincronización.
  - Semáforos — una variable protegida que provee control de acceso a un recurso en común.
  - Monitores — corresponde a un data type con operaciones definidas por el programador ejecutadas con mutua-exclusión.
  - Mutex (mutua-exclusión) — es una primitiva de sincronización que garantiza el acceso exclusivo a un recurso compartido.

Tecnologías de Construcción

- Middleware
  - Es una clasificación para todo software que provea servicios sobre la capa del sistema operativo y por debajo de la capa aplicación.
  - Puede proveer contenedores en tiempo de ejecución para componentes de software para proveer pasaje de mensajes, persistencia y ubicación transparente a través de la red.
  - El middleware orientado a mensajes moderno provee usualmente de un Enterprise Service Bus (ESB) que soporta la interacción orientada a servicios y la comunicación entre múltiples aplicaciones de software.

Tecnologías de Construcción

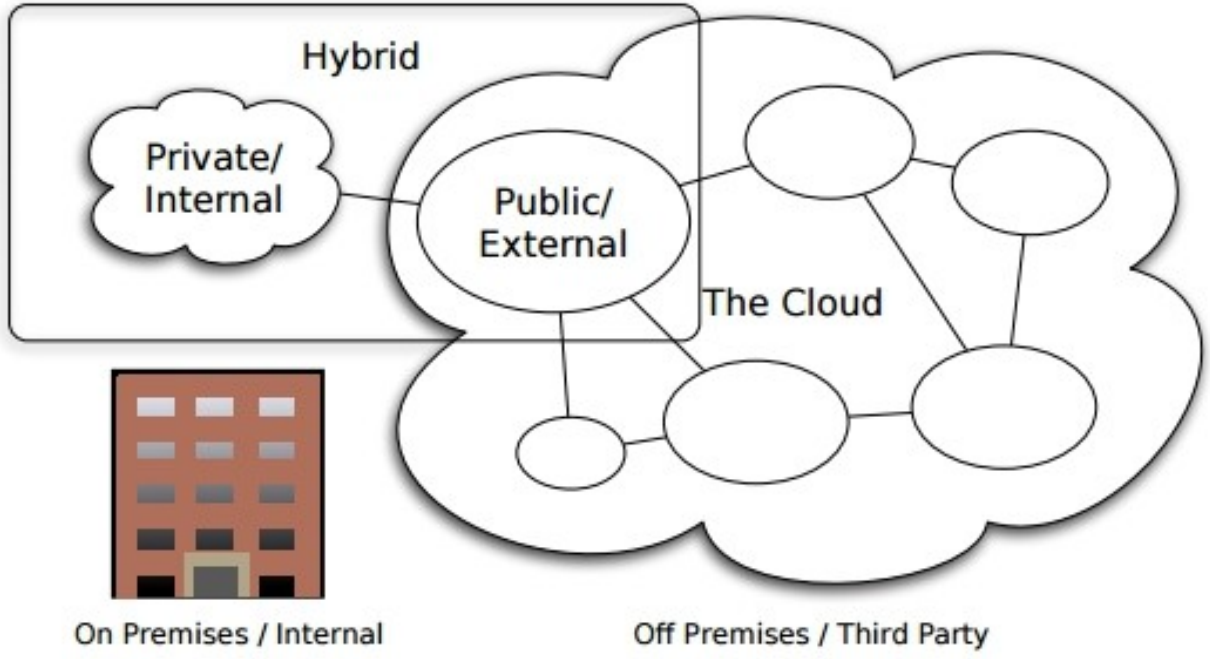
- Cloud Computing

  https://es.wikipedia.org/wiki/Computaci%C3%B3n_en_la_nube

  - Paradigma que permite ofrecer servicios de computación a través de una red, que usualmente es Internet.
  - Software como servicio (SaaS)
    - Una instancia del software que corre en la infraestructura del proveedor y sirve a múltiples organizaciones de clientes.
  - Plataforma como servicio (PaaS)
    - Es la encapsulación de una abstracción de un ambiente de desarrollo y el empaquetamiento de una serie de módulos o complementos que proporcionan, normalmente, una funcionalidad horizontal (persistencia de datos, autenticación, mensajería, etc.).

Tecnologías de Construcción

- Cloud Computing - Plataforma como servicio (PaaS)
  - Ejemplos: Google App Engine, Microsoft Azure, Bluemix de IBM.
  - En algunos modelos de servicio se ofrece la plataforma de desarrollo y las herramientas de programación por lo que puede desarrollar aplicaciones propias y controlar la aplicación, pero no controla la infraestructura.
  - **Tipos: Públicos, privados e híbridos**



Tecnologías de Construcción

- Métodos de construcción de software distribuido
  - Se distinguen por cuestiones de paralelismo, comunicación y tolerancia a fallas.
- Construcción de sistemas heterogéneos
  - Los sistemas heterogéneos consisten en una variedad de unidades computacionales especializadas de diferentes tipos. En general son independientes pero se comunican entre sí.
  - Se plantean cuestiones relacionadas a los distintos lenguajes utilizados, así como de aspectos sobre la comunicación.

Tecnologías de Construcción

- Cloud Computing
  - Infraestructura como servicio (IaaS)
    - Es un medio de entregar almacenamiento básico y capacidades de cómputo como servicios estandarizados en la red.
    - A veces llamados Hardware as aService (HaaS),
    - Por ejemplo, Amazon Web Services.
  - Aplicaciones: Dropbox, iCloud, OneDrive, Google Drive.
  - Ventajas / Desventajas.

Tecnologías de Construcción

- Análisis de Desempeño (Performance) y Puesta a punto (Tuning)
  - Análisis de desempeño — se estudia el comportamiento del programa en su ejecución.
  - Puesta a punto del código — involucra (en general) cambios menores para ajustar el desempeño del programa utilizando la información recolectada en el análisis de desempeño.
- Estándares de plataforma
  - Permiten que los programadores desarrollen aplicaciones portables que pueden ser ejecutadas en entornos compatibles sin realizar cambios. Por ej. J2EE.

Tecnologías de Construcción

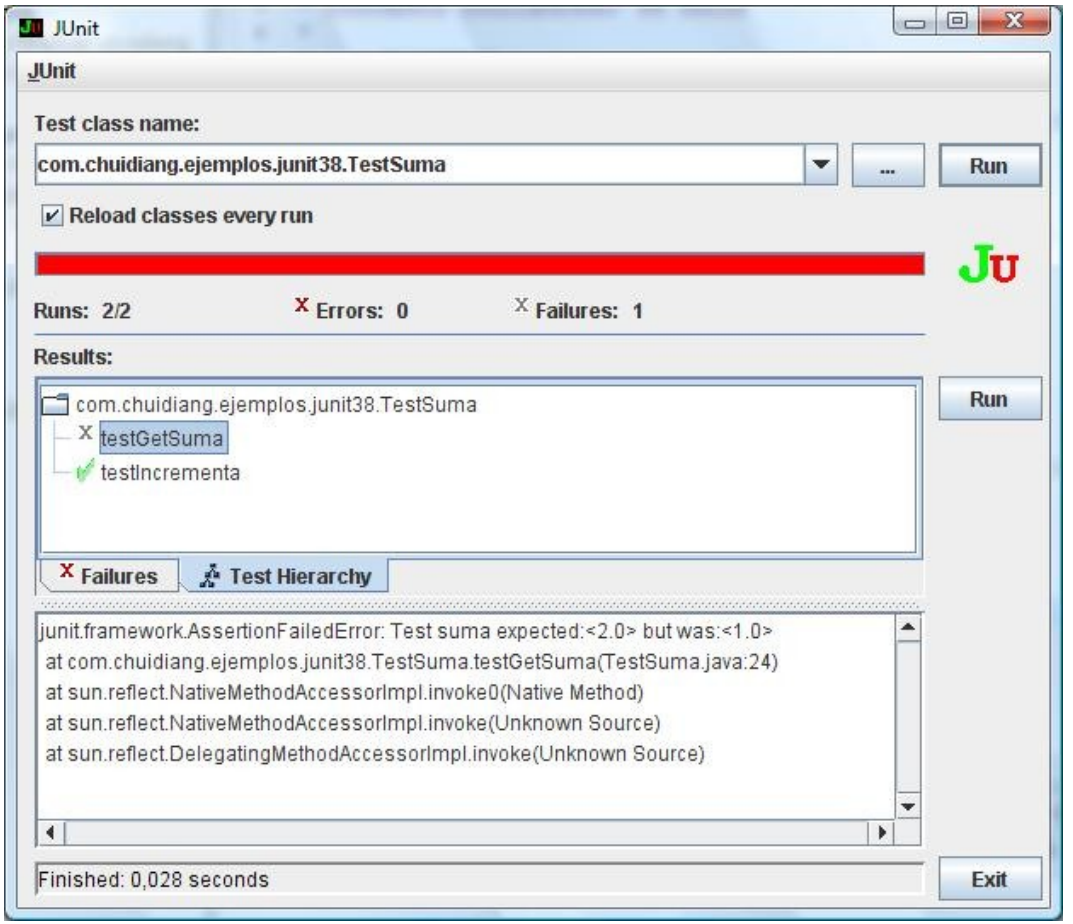
- Test-First Programming
  - O Test-driven development (TDD). Estilo de programación en el cual se escriben las pruebas antes que el código.

Herramientas de Construcción

- Entornos de desarrollo (IDEs)
  - Proveen facilidades a los programadores para desarrollar software. La elección de un IDE puede afectar la eficiencia y la calidad de la construcción del software.
- GUI Builders
  - Son herramientas que permiten el desarrollo y mantenimiento de interfaces de usuario gráficas. En general de un modo WYSIWYG (what you see is what you get – lo que ves es lo que obtienes).

Herramientas de Construcción

- Herramientas de pruebas unitarias
  - Permiten automatizar las pruebas unitarias. El programador especifica las entradas y salidas esperadas y la herramienta permite ejecutar un conjunto de pruebas indicando la salida obtenida.



Cuestiones variadas

Flow - Flujo

- El flujo es una condición de participación profunda, casi meditativa.
- Has estado en este estado a menudo, por lo que no tenemos que describírtelo.
- No todos los roles de trabajo requieren un estado de flujo para ser productivo, pero para cualquier persona involucrada en tareas de ingeniería, diseño, desarrollo, redacción o tareas similares, el flujo es imprescindible. Es solamente cuando estás en flujo, el trabajo va bien.
- Requiere unos 15 minutos para entrar en ese estado. Durante este período de inmersión, usted es particularmente sensible al ruido y la interrupción. Un entorno disruptivo puede dificultar o imposible de alcanzar el flujo.

Estándares de Programación

- Un estándar o estilo de programación son convenciones y buenas prácticas para escribir código fuente en determinados lenguajes de programación.

Estándares de Programación

- **Consideraciones generales**
  - Fuerte dependencia del **lenguaje** de programación
  - La mayoría del software es desarrollado/mantenido por **equipos**
  - Aunque se trabaje en forma individual, definir un estilo de codificación ayuda a organizarce
  - Aportan a la "**mantenibilidad**" del Software
  - Priorizar el código "legible"

Estándares de Programación

- **Nomenclatura** de
  - Constantes,
  - Variables,
  - Tipos de datos,
  - Procedimientos, Funciones
  - Clases, Módulos,
  - Controles, etc.

```
get a b c
if a < 24 and b < 60 and c < 60
  return true
else
  return false
```

```
get horas minutos segundos
if horas < 24 and minutos < 60 and segundos < 60
  return true
else
  return false
```

Estándares de Programación

- Identación del Código

```
if(horas < 24 && minutos < 60 && segundos < 60){
    return true;
}else{
    return false;
}
```

o bien:

```
if(horas < 24 && minutos < 60 && segundos < 60)
{
    return true;
}
else
{
    return false;
}
```

con algo como:

```
if(horas<24&&minutos<60&&segundos<60){return true;}
else{return false;}
```

Estándares de Programación

- Otras convenciones:
  - Mensajes de alerta/error
  - Manejo de Excepciones
  - Estructuras de Control (bucles, iteraciones)
  - Acceso a base de datos
  - Diseño de menús y atajos por teclado
  - Comentarios de código
  - Espacios y líneas en blanco
  - Operaciones de bloqueo deben incluir un timeout
  - Operaciones que toman tiempo deben mostrar indicador de avance o estado procesando

Reutilización de Código

- Administración de una *Base de Conocimiento* o *Biblioteca de Código*.

Roles:

- Productor
  - Identificar qué producir
  - Definir características

- Consumidor
  - Identificar qué puede servir
  - Evaluar si efectivamente sirve

Documentación del Código

- Interna
  - Documentación, comentarios internos al código fuente.

- Externa
  - Documentación técnica
  - Existen herramientas automáticas para generarla
  - Puede incluir:
    - Problema
    - Algoritmos
    - Datos
    - Interfaces

Documentación del Código

Documentación Interna

- Fundamental para el Mantenimiento

- Encabezamiento
  - descripción, nombre, programador

- Comentarios, ejemplos:

```
// Incremento i3
i3 = i3 + 1;

// ajusto contador para leer siguiente
i3 = i3 + 1;

case_counter = case_counter + 1;
```

Programación en Pares

- Es una práctica que reúne dos programadores a trabajar en forma conjunta
- Uno tiene el rol de “conductor” (el que escribe) y el otro de “acompañante” (el que revisa); o piloto y co-piloto.
- Su éxito depende de la combinación de las habilidades y personalidades del par.
- La mayor crítica a esta práctica es que es difícil aseverar si afecta en forma positiva o negativa a la productividad ¿usted que opina?

O'REILLY®

Becoming a Better Programmer

A HANDBOOK FOR PEOPLE WHO CARE ABOUT CODE

Pete Goodliffe

Debug

- *If debugging is the process of removing software bugs, then programming must be the process of putting them in. — Edsger Dijkstra*
- Determinar los pasos para reproducir un bug.
- Al encontrar un punto dónde sabemos que el sistema estaba bien y un lugar dónde falló tenemos un pedazo de código a revisar.
- → Usar aserciones o revisar invariantes, por ej. imprimiendo mensajes (mejor aserciones o herramientas del IDE).
- → búsqueda binaria, para dividir el pedazo de código a la mitad.
- → arqueología de software
- Luego análisis causal y aprender. Usar debuggers!!

Debug

- Si el bug no es reproducible
- → llevar registro del bug, contexto, consecuencias para buscar patrones en el futuro.
- → considerar agregar aserciones o logs para registrar más información.
- → si es un problema grave buscar tener un ambiente lo más parecido al cual dio problemas.