1. A01. O software e os proxectos de desenvolvemento de software

1.1 Introdución

Na actividade que nos ocupa preténdense os seguintes obxectivos:

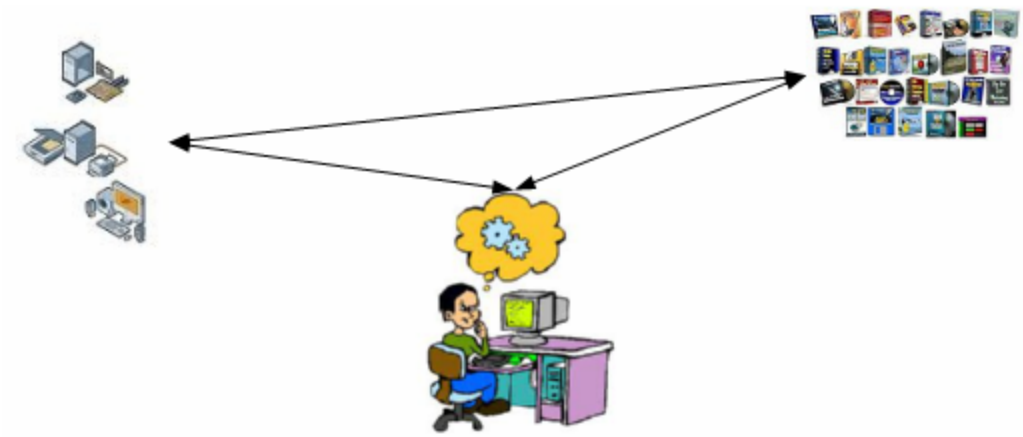
- Identificar algoritmo, software, versións, software de aplicación, software de sistema, hardware e recoñecer a relación entre eles.
- Identificar e caracterizar as fases de desenvolvemento de software no modelo en cascada, en espiral, na programación extrema e en Métrica 3.

1.2 Actividade

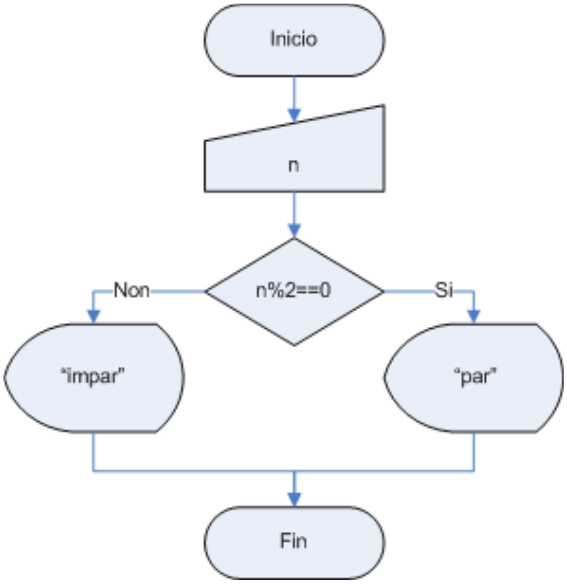
Software

A Real Academia Galega define a informática como a "Ciencia do tratamento automático da información por medio de máquinas electrónicas". Este tratamento necesita de:

- Soporte físico ou hardware formado por todos os compoñentes electrónicos tanxibles involucrados no tratamento. Segundo a Real Academia Galega é máis aconsellable dicir soporte físico que hardware e defíneo como a maquinaria ou elementos que compoñen un ordenador.
- Soporte lóxico ou software que fai que o hardware funcione e está formado por todos os compoñentes intanxibles involucrados no tratamento: programas, datos e documentación. . Segundo a Real Academia Galega é máis aconsellable dicir soporte lóxico que software e defíneo como o conxunto de ordes e programas que permiten utilizar un ordenador.
- Equipamento humano ou persoal informático que manexa o equipamento físico e o lóxico para que todo o tratamento se leve a cabo.



O concepto de programa informático está moi ligado ao concepto de algoritmo. Un algoritmo é un conxunto de instrucións ou regras ben definidas ordenadas y finitas que permite resolver un problema. Exemplo de algoritmo representado mediante un diagrama de fluxo que permite ver se un número tecleado é par ou impar e que considera o 0 como número par:



O algoritmo pode escribirse nunha linguaxe de programación utilizando algunha ferramenta de edición, dando lugar a un código que deberá de gravarse nunha memoria externa non volátil para que perdure no tempo. Exemplo de código escrito en linguaxe Java que se corresponde co anterior algoritmo:

```
package parimpar;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        short n;
        Scanner teclado = new Scanner(System.in);
        System.out.printf("Teclee o número enteiro entre %d e %d:",
      Short.MIN_VALUE, Short.MAX_VALUE);
        n = teclado.nextShort();
        if(n%2==0){
            System.out.printf("%d é par\n",n);
        }
        else{
            System.out.printf("%d é impar\n",n);
        }
    }
}
```

O código terá que sufrir algunhas transformacións para que finalmente se obteña un programa que poida ser executado nun ordenador. Para a execución é preciso que o programa estea almacenando na memoria interna e volátil do ordenador; así o procesador pode ir collendo cada orde que forma o programa, resolvéndoa e controlando a súa execución. Se é preciso, accede á memoria interna para manipular variables, ou aos periféricos de entrada, saída ou almacenamento, fai cálculos ou resolve expresións lóxicas ata que finaliza con éxito a última instrución ou se atopa cun erro irresoluble.

Na execución do programa correspondente ao código exemplo anterior, o procesador necesitaría un número enteiro subministrado a través do teclado (dispositivo ou periférico de entrada), obtería o resultado dunha operación aritmética (resto da división enteira de n entre 2), resolvería unha expresión lóxica (descubrir se o resto anterior é 0), e executaría a instrución alternativa para que no caso de que o resto sexa 0 saia pola pantalla (dispositivo ou periférico de saída) que n é par ou que é impar. Exemplo de execución do código anterior e aspecto do resultado da execución na pantalla de texto cando se teclea o número 11:

```
run:
Teclee o número enteiro entre -32768 e 32767:11
11 é impar
BUILD SUCCESSFUL (total time: 7 seconds)
```

Os programas informáticos son imprescindibles para que os ordenadores funcionen e son conxuntos de instrucións que unha vez executadas realizarán unha ou varias tarefas. O software é un conxunto de programas e neste concepto inclúense tamén os datos que se manexan e a documentación deses programas.

Clasificación de software

Pódese clasificar o software en dous tipos:

- Software de sistema. Controla e xestiona o hardware facendo que funcione, ocultando ao persoal informático os procesos internos deste funcionamento. Exemplos deste tipo de software son:
  - Sistemas operativos
  - Controladores de dispositivos
  - Ferramentas de diagnóstico
  - Servidores (correo, web, DNS,...)
  - Utilidades do sistema (compresión de información, rastreo de zoas defectuosas en soportes,...)
- Software de aplicación. Permite levar a cabo unha ou varias tarefas especificas en calquera campo de actividade dende que está instalado o software básico de sistema. Exemplos deste tipo de software son:
  - Aplicacións para control de sistemas e automatización industrial
  - Aplicacións ofimáticas
  - Software educativo
  - Software empresarial: contabilidade, nóminas, almacén,...
  - Bases de datos
  - Videoxogos
  - Software de comunicacións: navegadores web, clientes de correo,...
  - Software médico
  - Software de cálculo numérico
  - Software de deseño asistido por ordenador
  - Software de programación que é o conxunto de ferramentas que permiten ao programador desenvolver programas informáticos como por exemplo:

- Editores de texto
- Compiladores
- Intérpretes
- Enlazadores
- Depuradores
- Contornos de desenvolvemento integrado (IDE) que agrupa as anteriores ferramentas nun mesmo contorno para facilitar ao programador a tarefa de desenvolver programas informáticos e que teñen unha avanzada interface gráfica de usuario (GUI).

Dentro do software de aplicación pódese facer unha división entre:

- Software horizontal ou xenérico que se poden utilizar en diversos contornos como por exemplo as aplicacións ofimáticas.
- Software vertical ou a medida que só se poden utilizar nun contorno específico como por exemplo a contabilidade feita a medida para unha empresa en concreto.

Tarefa 1. Buscar en internet nomes comerciais de software de sistema e software de aplicación.

Desenvolvemento de software

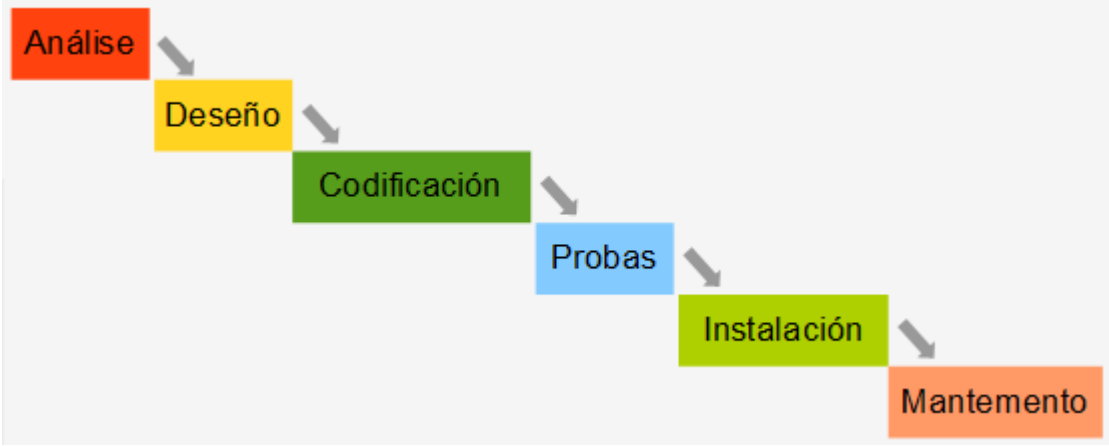
O proceso de desenvolvemento de software é moi diferente dependendo da complexidade do software. Por exemplo, para desenvolver un sistema operativo é necesario un equipo disciplinado, recursos, ferramentas, un proxecto a seguir e alguén que xestione todo. No extremo oposto, para facer un programa que visualice se un número enteiro que se teclea é par ou impar, só é necesario un programador ou un afeccionado á programación e un algoritmo de resolución.

Fritz Bauer nunha reunión do Comité Científico da OTAN en 1969 propuxo a primeira definición de Enxeñaría de software como o establecemento e uso de principios de enxeñería robustos, orientados a obter economicamente software que sexa fiable e funcione eficientemente sobre máquinas reais. Posteriormente, moitas personalidades destacadas do mundo do software matizaron esta definición que se normalizou na norma IEEE 1993 que define a Enxeñaría de software como a aplicación dun enfoque sistemático, disciplinado e medible ao desenvolvemento, funcionamento e mantemento do software.
Durante décadas os enxeñeiros e enxeñeiras de software foron desenvolvendo e mellorando paradigmas (métodos, ferramentas e procedementos para describir un modelo) que foran sistemáticos, predicibles e repetibles e así mellorar a produtividade e calidade do software.
A Enxeñaría de software é imprescindible en grandes proxectos de software, debe de ser aplicada en proxectos de tamaño medio e recoméndase aplicar algún dos seus procesos en proxectos pequenos.
Existen moitos modelos a seguir para desenvolver software. O máis clásico é o modelo en cascada. Destacan entre todos o modelo en espiral baseado en prototipos, a programación extrema como representativo dos métodos de programación áxil, e a Métrica 3 como modelo a aplicar en grandes proxectos relacionados coas institucións públicas.

Paradigma de ciclo de vida clásico ou modelo en cascada

O paradigma de ciclo de vida clásico do software, tamén chamado modelo en cascada consta das fases: análise, deseño, codificación, probas, instalación e mantemento e a documentación é un aspecto implícito en todas as fases. Algúns autores nomean estas fases con nomes lixeiramente diferentes, ou poden agrupar algunhas para crear unha nova fase ou nomear unha fase con máis detalle, pero en esencia as fases son as mesmas. Algunhas destas fases tamén se utilizan noutros modelos con lixeiras variantes.

As fases vanse realizando de forma secuencial utilizando a información obtida ao finalizar unha fase para empezar a fase seguinte. Deste xeito o cliente non pode ver o software funcionando ata as últimas fases e daría moito traballo corrixir un erro que se detecta nas últimas fases pero que afecta ás primeiras.



Análise

Nesta fase o analista captura, analiza e especifica os requisitos que debe cumprir o software. Debe obter a información do cliente ou dos usuarios do software mediante entrevistas planificadas e habilmente estruturadas nunha linguaxe comprensible para o usuario. O resultado desta captura de información depende basicamente da habilidade e experiencia do analista aínda que poida utilizar guías ou software específico.

Ao finalizar esta fase debe existir o documento de especificación de requisitos do software (ERS), no que estarán detallados os requisitos que ten que cumprir o software, debe valorarse o custo do proxecto e planificarse a duración do mesmo. Toda esta información ten que comunicarse ao cliente para a súa aceptación.

A linguaxe utilizada para describir os ERP pode ser descritiva ou máis formal e rigorosa utilizando casos de usos na linguaxe de modelado UML. O estándar IEEE 830-1998 recolle unha norma de prácticas recomendadas para a especificación de requisitos de software.

Deseño

Nesta fase o deseñador deberá de descompoñer e organizar todo o sistema software en partes que podan elaborarse por separado para así aproveitar as vantaxes do desenvolvemento de software en equipo.

O resultado desta fase plásmase no documento de deseño de software (SDD) que contén a estrutura global do sistema, a especificación do que debe facer cada unha das partes e a maneira de combinarse entre elas e é a guía que os programadores e probadores de software deberán ler, entender e seguir. Este documento incluirá o deseño lóxico de datos, o deseño da arquitectura e estrutura, o deseño dos procedementos, a organización do código fonte e a compilación, e o da

interface entre o home e o software. Exemplos de diagramas que indican como se construirá o software poden ser: modelo E/R para os datos, diagramas UML de clases, diagramas UML de despregue e diagramas UML de secuencia.
Nesta fase debe tratarase a seguridade do proxecto mediante unha análise de riscos (recompilación de recursos que deben ser protexidos, identificación de actores e roles posibles, recompilación de requisitos legais e de negocio como encriptacións ou certificacións a cumprir, etcétera) e a relación de actividades que mitigan eses riscos.

Codificación

Esta fase tamén se chama fase de programación ou implementación[1]. Nela o programador transforma o deseño lóxico da fase anterior a código na linguaxe de programación elixida, de tal forma que os programas resultantes cumpran os requisitos da análise e poidan ser executado nunha máquina.

Mentres dura esta fase, poden realizarse tarefas de depuración do código ou revisión inicial do mesmo para detectar erros sintácticos, semánticos e de lóxica.

Probas

Esta fase permite aplicar métodos ou técnicas ao código para determinar que tódalas sentencias foron probadas e funcionan correctamente.

As probas teñen que planificarse, deseñarse, executarse e avaliar os resultados. Considérase que unha proba é boa se detecta erros non detectados anteriormente. As probas realizadas inmediatamente despois da codificación poden ser:

- Probas unitarias cando permiten realizar tests a anacos pequenos de código cunha funcionalidade específica.
- Probas de integración cando permiten realizar tests a un grupo de anacos de código que superaron con éxito os correspondentes tests unitarios e que interactúan entre eles.

Outras probas sobre o sistema total poden ser:

- Probas de validación ou aceptación para comprobar que o sistema cumpre os requisitos do software especificados no ERS.
- Probas de recuperación para comprobar como reacciona o sistema fronte a un fallo xeral e como se recupera do mesmo.
- Probas de seguridade para comprobar que os datos están protexidos fronte a agresións externas.
- Probas de resistencia para comprobar como responde o sistema a intentos de bloqueo ou colapso.
- Probas de rendemento para someter ao sistema a demandas do usuario extremas e comprobar o tempo de resposta do sistema.

---

[1] Segundo a Real Academia Española da lingua, implementar é poñer en funcionamento, aplicar métodos, medidas, etc., para levar algo a cabo. En informática esta palabra aplícase en varios contextos como por exemplo os seguintes textos extraídos das páxinas oficiais:

- "Mozilla Firefox respecta na súa implementación as especificacións de W3C" para indicar que o código de Mozilla cumpre esas especificacións.
- "Microsoft Visual Studio permite crear proxectos de implementación e instalación que permiten distribuír unha aplicación finalizada para a súa instalación noutros equipos" para indicar que permite planear, instalar e manter unha aplicación finalizada.

As probas deberán de ser realizadas en primeiro lugar polos creadores do software pero é recomendable que tamén sexan realizadas por especialistas que non participaron na creación e finalmente sexan realizadas por usuarios.

O software pode poñerse a disposición dos usuarios cando aínda no está acabado e entón noméase co nome comercial e un texto que indica o nivel de acabado. Ese texto pode ser:

- Versión Alfa. Versión inestable, á que aínda se lle poden engadir novas características.
- Versión Beta. Versión inestable á que non se lle van a engadir novas características pero que pode ter erros.
- Versión RC (Release Candidate) é case a versión final pero aínda poden aparecer erros.
- Versión RTM (Release To Manufacturing). Versión estable para comercializar.

Tarefa 2. Buscar en internet aplicacións software dispoñibles para o usuario en versión alfa, beta, RC ou RTM.

Instalación

Esta fase tamén se denomina despregue ou implantación e consiste en transferir o software do sistema ao ordenador destino e configuralo para que poida ser utilizados polo usuario final. Esta fase pode consistir nunha sinxela copia de arquivos ou ser máis complexo como por exemplo: copia de programas e de datos que están comprimidos a localizacións específicas do disco duro, creación de accesos directos no escritorio, creación de bases de datos en localizacións específicas, etcétera.

Existen ferramentas software que permiten automatizar este proceso que se chaman instaladores.
No caso dunha instalación simple, pode ocorrer que o instalador xere uns arquivos que permitan ao usuario final facer de forma automática unha instalación guiada e sinxela; noutro caso a instalación ten que ser feita por persoal especializado.
O software pode pasar a produción despois de resolver o proceso de instalación, é dicir, pode ser utilizado e explotado polo cliente.

Mantemento

Esta fase permite mellorar e optimizar o software que está en produción. O mantemento permitirá realizar cambios no código para corrixir erros encontrados, para facer o código máis perfecto (optimizar rendemento e eficacia, reestruturar código, perfeccionar documentación, etcétera), para que evolucione (agregar, modificar ou eliminar funcionalidades segundo necesidades do mercado, etcétera), ou para que se adapte (cambios no hardware utilizado, cambios no software do sistema xestor de bases de datos, cambios no sistema de comunicacións, etcétera).

Ao redor do 2/3 partes do tempo invertido en Enxeñería de software está dedicado a tarefas de mantemento e ás veces estas tarefas son tantas e tan complexas que é menos custoso volver a deseñar o código.
As versións de software resultantes do mantemento noméanse de diferente maneira dependendo do fabricante. Por exemplo: Debian 7.6, NetBeans IDE 6.5, NetBeans IDE 7.0.1, NetBeans 8.0, Java SE 8u20 (versión 8 update 20). Algúns fabricantes subministran aplicación que son parches que melloran o software instalado como por exemplo Service Pack 1 (SP1) para Windows Server 2008 R2, ou Service Pack 2 (SP2) para Windows Server 2008 R2.

Tarefa 3. Buscar en internet as últimas versións de Debian, NetBeans, Java, Windows.

Documentación

A creación de documentación está asociada a tódalas fases anteriores e en especial ás fases de codificación, probas e instalación. Para estas últimas fases pódese distinguir entre:

- Documentación interna. É a que aparece no código dos programas para que os programadores encargados de mantelo poidan ter información extra sen moito esforzo e para que de forma automática poida extraerse fóra do código esa información nun formato lexible, como por exemplo HTML, PDF, CHM, etcétera.

  Exemplo de código Java con comentarios Javadoc que aportan información extra e permiten que algunhas aplicacións xeren automaticamente unha páxina web coa información que extraen dos comentarios:

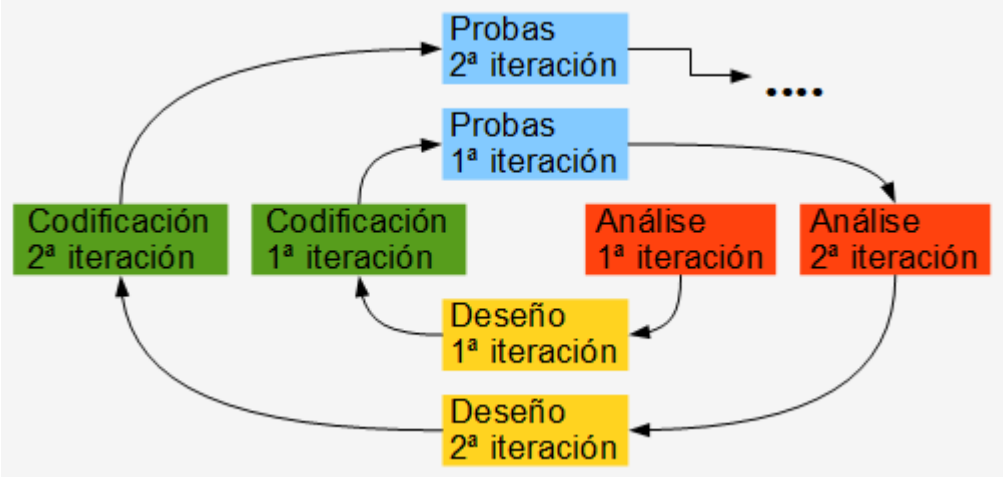
```
/** Este método calcula el número de reintentos de una
    determinada opción.
    @param opcion Cadena de texto con la opción a parsear
           para extraer el número de reintentos. La sintaxis es
           la siguiente: "reintentos:XXX", donde XXX será una
           cadena de texto que se interpretará como un
           número (valor retornado).
    @param similitud Flag que si vale true (valor por defecto)
           hará que el método no distinga las letras mayúsculas
           a la hora de intentar parsear la sintaxis, de modo
           que "reintentos:XXX" sea igual de válido que
           "ReinteNTos:XXX".
    @return Devuelve el número de reintentos o -1 en caso
            de error. */
int CalcularReintentos (const char *opcion,  bool similitud = true);
```

- Documentación externa que é a que se adxunta aos programas e pode estar dirixida:
  - Aos programadores. Formada por exemplo por: código fonte, explicación de algoritmos complicados, especificación de datos e formatos de entrada/saída, listado de arquivos que utiliza ou xera a aplicación, linguaxe de programación, descrición de condicións ou valores por defecto que utiliza, diagramas de deseño..
  - Aos usuario. Formada por exemplo por: requisitos do sistema (tipo de ordenador no que funciona, sistema operativo que require, recursos hardware necesarios, etcétera), detalles sobre a instalación, explicacións para utilizar o software de forma óptima, descrición de posibles errores de funcionamento ou instalación e a forma de corrixilos.
- Autodocumentación que é documentación á que se accede durante a execución dos programas e pode conter: índice de contidos, guías sobre o manexo do programa, asistentes, axuda contextual, autores dos programas, versións dos mesmos, direccións de contacto, enlaces de referencia..

Modelo en espiral

Este modelo baséase na creación dun prototipo do proxecto que se vai perfeccionando en sucesivas iteracións a medida que se engaden novos requisitos, pasando en cada iteración

polo proceso de análise, deseño, codificación e probas descritos no modelo en cascada. Ao final de cada iteración o equipo que desenvolve o software e o cliente analizaran o prototipo conseguido e acordarán se inician unha nova iteración. Sempre se traballa sobre un prototipo polo que no momento que se decida non realizar novas iteracións e acabar o produto, haberá que refinar o prototipo para conseguir a versión final acabada, estable e robusta.



Programación eXtrema[2]

A programación eXtrema ou *eXtreme Programming* é un método de desenvolvemento áxil de software baseado en iteracións sobre as fases de planificación, deseño, codificación e probas.

Planificación

Cada reunión de planificación é coordinada polo xestor do proxecto e nela:

- O cliente indica mediante frases curtas as prestacións que debe ter o software sen utilizar ningún tipo de tecnicismos nin ferramenta especial.
- Os desenvolvedores de software converterán cada prestación en tarefas que duren como máximo tres días ideais de programación de tal xeito que a prestación completa non requira máis de 3 semanas. De non conseguir estas cifras, revisaríanse as prestación e as tarefas de acordo co desexo do cliente.
- Entre todos decídese o número de prestacións que formarán parte de cada iteración que se denomina velocidade do proxecto.
- Ao final de cada iteración farase unha reunión de planificación para que o cliente valore o resultado; se non o acepta, haberá que engadir as prestacións non aceptadas á seguinte iteración e o cliente deberá de reorganizar as prestacións que faltan para que se respecte a velocidade do proxecto.

É importante a mobilidade das persoas, é dicir, que en cada iteración os desenvolvedores traballen sobre partes distintas do proxecto, de forma que cada dúas ou tres iteracións, os desenvolvedores haxan traballado en todas as partes do sistema.

[2] AYCART PÉREZ, David. GIBERT GINESTA, Marc HERNÁNDEZ MATÍAS, Martín, MAS HERNÁNDEZ, Jordi. *Ingeniería de software en entornos de SL*. Universitat Ouberta de Catalunya.

Deseño

Á diferenza do modelo en cascada, nesta fase utilízase unha tarxeta manual tipo CRC (*class, responsabilities, collaboration*) por cada obxecto do sistema, na que aparece o nome da clase, nome da superclase, nome das subclases, responsabilidades da clase, e obxectos cos que colabora. As tarxetas vanse colocando riba dunha superficie formando unha estrutura que reflicta as dependencias entre elas. As tarxetas vanse completando e recolocando de forma manual a medida que avanza o proxecto. Os desenvolvedores reuniranse periodicamente e terán unha visión do conxunto e de detalle mediante as tarxetas.

Codificación e probas

Unha diferenza desta fase con relación á fase de codificación do modelo en cascada é que os desenvolvedores teñen que acordar uns estándares de codificación (nomes de variables, sangrías e aliñamentos, etcétera), e cumprilos xa que todos van a traballar sobre todo o proxecto.

Outra diferenza é que se aconsella crear os test unitarios antes que o propio código a probar xa que entón se ten unha idea máis clara do que se debe codificar.
Unha última diferenza é que se aconsella que os programadores desenvolvan o seu traballo por parellas (*pair programming*) xa que está demostrado que dous programadores traballando conxuntamente fronte ao mesmo monitor pasándose o teclado cada certo tempo, fano ao mesmo ritmo que cada un polo seu lado pero o resultado final é de moita máis calidade xa que mentres un está concentrado no método que está codificando, o outro pensa en como ese método afecta ao resto de obxectos e as dúbidas e propostas que xorden reducen considerablemente o número de erros e os problemas de integración posteriores.

Métrica v.3

Métrica versión 3 é unha metodoloxía de planificación, desenvolvemento e mantemento de sistemas de información promovido pola *Secretaría de Estado de Administraciones Públicas* do *Ministerio de Hacienda y Administraciones Públicas* e que cubre o desenvolvemento estruturado e o orientado a obxectos .

Esta metodoloxía ten como referencia o Modelo de Ciclo de Vida de Desenvolvemento proposto na norma ISO 12.207 "*Information technology- Software live cycle processes*".
Consta de tres procesos principais: planificación, desenvolvemento e mantemento. Cada proceso divídese en actividades non necesariamente de execución secuencial e cada actividade en tarefas.
No portal de administración electrónica (http://administracionelectronica.gob.es/) pódese acceder aos documentos pdf no que está detallada toda a metodoloxía e o persoal informático que intervén en cada actividade.

Planificación

O proceso de planificación de sistemas de información (PSI) ten como obxectivo a obtención dun marco de referencia para o desenvolvemento de sistemas de información que respondan a obxectivos estratéxicos da organización e é fundamental a participación da alta dirección da organización. Consta das actividades:

- Descrición da situación actual.
- Un conxunto de modelos coa arquitectura da información.

- Unha proposta de proxectos a desenvolver nos próximos anos e a prioridade de cada un.
- Unha proposta de calendario para a execución dos proxectos.
- A avaliación dos recursos necesarios para desenvolver os proxectos do próximo ano.
- Un plan de seguimento e cumprimento de todo o proposto.

Desenvolvemento

Cada proxecto descrito na planificación ten que pasar polo proceso de desenvolvemento de sistemas de información que consta das actividades:

- Estudio de viabilidade do sistema (EVS) no que se analizan os aspectos económicos, técnicos, legais e operativos do proceso e se decide continuar co proceso ou abandonalo. No primeiro caso haberá que describir a solución encontrada: descrición, custo, beneficio, riscos, planificación da solución. A solución pode ser desenvolver software a medida, utilizar software estándar de mercado, solución manual ou unha combinación delas.
- Análise do sistema de información (ASI) para obter a especificación de requisitos software que conterá as funcións que proporcionará o sistema e as restricións ás que estará sometido, para analizar os casos de usos, as clases e interaccións entre elas, para especificar a interface de usuario, e para elaborar o plan de probas.
- Deseño do sistema de información (DSI) no que se fai o deseño de comportamento do sistema para cada caso de uso, o deseño da interface de usuario, o deseño de clase, o deseño físico de datos (se é necesario tamén se fai o deseño da migración e carga inicial de datos), a especificación técnica do plan de probas e o establecemento dos requisitos de implantación (implantación do sistema, formación de usuarios finais, infraestruturas, etcétera).
- Construción do sistema de información (CSI) no que se prepara a base de datos física, se prepara o entorno de construción, xérase o código, execútanse as probas unitarias, as de integración e as de sistema, elabóranse os manuais de usuario, defínese a formación dos usuarios finais e constrúense os compoñentes e procedementos da migración e carga inicial de datos.
- Implantación e aceptación do sistema (IAS) que ten como obxectivo a entrega e aceptación do sistema total e a realización de todas as actividades necesarias para o paso a produción. Para iso séguense os pasos: formar ao equipo de implantación, formar aos usuarios finais, realizar a instalación, facer a migración e carga inicial de datos, facer as probas de implantación (comprobar que o sistema funcione no entorno de operación), facer as probas de aceptación do sistema (comprobar que o sistema cumpre os requisitos iniciais do sistema), establecer o nivel de mantemento e servizo para cando o produto estea en produción. O último paso é o paso a produción para o que se analizarán os compoñentes necesarios para incorporar o sistema ao entorno de produción, de acordo ás características e condicións do entorno no que se fixeron as probas e se realiza a instalación dos compoñentes necesarios valorando a necesidade de facer unha nova carga de datos, unha inicialización ou unha restauración, fíxase a data de activación do sistema e a eliminación do antigo.

Mantemento

O obxectivo deste proceso é a obtención dunha nova versión do sistema de información desenvolvido con Métrica 3, a partir das peticións de mantemento que os usuarios realizan con motivo de problemas detectados no sistema ou pola necesidade de mellora do mesmo.

Tarefa 4. Buscar en internet nomes de metodoloxías áxiles de desenvolvemento de software.