# Guía para la documentación de proyectos de software

Organización de Computadoras
Universidad Nacional del Sur
2017

## Estructura y contenido

### 1. Definiciones y especificación de requerimientos

Los requerimientos/requisitos de un sistema describen los servicios que ha de ofrecer el sistema y las restricciones asociadas a su funcionamiento, es decir, las propiedades o restricciones que deben satisfacerse, determinadas de forma precisa.

En este apartado se brindarán tres aspectos informativos:

a) ***Definición general del proyecto de software***: explicar en qué consiste el sistema o desarrollo en cuestión, cuál es la idea general y la funcionalidad principal del proyecto de software, así como también los propósitos y objetivos del desarrollo.
b) ***Especificación de requerimientos del proyecto***: incluir el detalle de los requerimientos técnicos y generales del mismo (por ejemplo, en el caso de un proyecto de software de la UNS, las consignas y pautas del mismo), los alcances y limitaciones de la implementación realizada. Deberá aclararse si el proyecto de software forma parte de algún sistema ya desarrollado; de ser el caso, especificar si se desarrolló una nueva versión o es una derivación.
c) ***Procedimientos* de instalación y prueba**: detallar cómo se realiza la obtención, instalación y/o prueba del sistema, junto las especificaciones generales de la plataforma o el entorno sobre el cual el software debe ser ejecutado.

***De la definición general del proyecto de software***

En la definición del proyecto se deberá brindar detalle sobre los puntos que se definen a continuación:

- Idea general: la funcionalidad principal del sistema *(qué..?)*
- Objetivos: los objetivos del desarrollo, y la necesidad cubierta por el sistema en cuestión *(para qué..?)*

- Usuarios: las personas o entidades que utilizarán el sistema o parte de él, y el nivel de experiencia del usuario hacia el cual el presente informe está dirigido *(quién..?)*

***De las especificación de requerimientos del proyecto de software***

Dentro de la especificación de requerimientos se dará información del proyecto de software sobre los siguientes puntos:

- Requisitos generales: las pautas y consignas que sigue el proyecto de software.
- Requisitos funcionales: los servicios que el sistema proporciona, las tareas que éste desarrolla.
- Información de autoría y *Legacy* del proyecto: explicitar si el proyecto de software forma parte de desarrollos previos/preexistentes o si es original, y en el caso correspondiente, detalles de retro-compatibilidad.
- Alcances del sistema: las limitaciones y alcances del desarrollo, de acuerdo a los objetivos previamente establecidos.

***De las especificaciones de procedimientos***

Dentro de la información relativa a procedimientos se distinguirá:

*Procedimientos de desarrollo:*

- Herramientas utilizadas: entornos de desarrollo integrados, plataformas y herramientas empleadas en la implementación del sistema.
- Planificación: una descripción global de la metodología utilizada para encarar y resolver el problema; por ejemplo: los pasos ejecutados a lo largo de la resolución del proyecto, a grandes rasgos.

*Procedimientos de instalación y prueba:*

- Requisitos no funcionales: si las hubiere, restricciones que afectan el normal desempeño del sistema.
- Obtención e instalación: una guía sencilla que explique el procedimiento básico para obtener e instalar el sistema. La guía debe estar dirigida a usuarios con nivel de *experiencia* preestablecido en la ***definición general del proyecto***.
- Especificaciones de prueba y ejecución: datos técnicos sobre la plataforma y/o entornos a utilizar en la prueba o ejecución del software en cuestión.

## 1. Arquitectura del sistema

Incluso un software de tamaño pequeño consta de la composición de varios módulos o partes interconectados de alguna forma. La descripción de la arquitectura del sistema informa sobre cuáles son estas partes, qué rol tienen dentro del software y la forma en que estas se organizan e interconectan.

La información sobre la arquitectura debería incluir como mínimo:

- **Descripción jerárquica:** Indica de qué forma se organizan jerárquicamente los componentes del sistema. Es decir, indicar si los mismos están organizados en paquetes, espacios de nombres o bien si el software posee una estructura monolítica.

- **Diagrama de módulos:** Consiste en un diagrama donde se representan todas las partes que componen el sistema y las relaciones que existen entre estas. El objetivo de este diagrama consiste en presentar una perspectiva global de la arquitectura y los componentes del sistema, no debería contener detalles técnicos sobre los módulos o las conexiones entre estos.

- **Descripción individual de los módulos:** Para cada módulo o parte del sistema, se debería realizar una breve descripción del mismo, la cual debería incluir mínimamente:

    - **Descripción general y propósito:** *¿qué es y qué debería hacer el módulo?*
    - **Responsabilidad y restricciones:** *¿cuál es su función específica dentro del sistema? ¿qué cosas puede y no puede hacer?*
    - **Dependencias:** Indicar cuales son los requisitos del módulo, es decir se debe contestar a preguntas tales como *¿qué necesita o requiere el módulo para funcionar? ¿necesita de servicios brindados por otros módulos o por librerías externas?*
    - **Implementación:** indicar en qué archivo o archivos se encuentra la implementación del módulo.

  No es el objetivo de esta sección dar detalles de *cómo* se realiza la implementación de los módulos, sino únicamente dar una idea general de *para qué* existe el módulo dentro del sistema.

- **Dependencias externas:** Si el software utiliza librerías o servicios externos estos deben listarse junto con una breve descripción de las mismas.

Adicionalmente en esta sección se deben listar los **aspectos técnicos** o **tecnologías empleadas** en el proyecto, tales como el lenguaje de programación, *frameworks*, librerías, etc.
Puede resultar de utilidad incluir junto a estos una breve descripción de las decisiones de diseño asociadas que llevaron a elegir la o las tecnologías en particular, es decir responder a *¿por qué se utilizó esta tecnología y no otra?*

1. **Diseño del modelo de datos**

Distinguir cuáles son las entidades involucradas en el sistema y mencionarlas en un formato *language-agnostic*.
Puede ser un diseño orientado a objetos, relacional, etc., lo importante es tener una idea general del modelo de datos: entidades, atributos y las relaciones entre ellas. Para ello es imprescindible incluir diagramas o gráficos que ayuden a visualizar el modelo de datos.

Un programa, aplicación o librería puede a su vez trabajar con varios tipos de datos:

- Datos de entrada.
- Datos internos.
- Datos de salida.

Distinguir claramente cada uno de ellos y describir su modelo.

1. **Descripción de procesos y servicios ofrecidos por el sistema**

Mencionar cuáles son los servicios o tareas que el sistema ofrece/implementa, y describir los procesos que realizan, para entender cómo funcionan, y cómo se pueden invocar/utilizar.

Para este propósito es conveniente incluir pseudo-algoritmos, diagramas de flujo, etc.

Tener presente que la descripción del proceso no significa mostrar el código, ni consiste tampoco brindar detalles específicos de cómo lo hace (funciones utilizadas para hacer cierta tarea) sino de explicar brevemente qué hace o cuál es su propósito. Se espera también una descripción de los datos de entrada y salida (Cantidad de argumentos, tipo y significado de cada uno).

En este punto es imprescindible que el código fuente de la aplicación esté enriquecido con comentarios. Estos deben conformar la documentación básica de todo proyecto, y a partir de los mismos debería poder construirse la descripción de alto nivel del funcionamiento de los procesos y servicios del sistema, así como sus funciones, subrutinas, módulos, clases, etc.

2. **Documentación técnica - Especificación API**

   Se indica el propósito y breve descripción de cada método/función, con su prototipo indicando argumentos (nombre, tipo, propósito de cada uno) y respuesta (tipo, descripción).

   Para llevar a cabo esta tarea, es posible utilizar una variedad de herramientas de generación de documentación automática, a partir del código en el encabezado de cada función (por ejemplo Javadoc, PHPDoc, Doxygen, etc).

   La documentación técnica debe pensarse como el manual del programador, y debe apuntar a aquellas personas que estarán a cargo de mantener, ampliar, o crear un proyecto derivado a partir de nuestro proyecto.

## Aspectos relevantes

- Indicar claramente cómo invocar el programa (signatura del programa completa, como la que haría cualquier sinopsis de una página de manual), conteniendo qué parámetros son opcionales, cuales son obligatorios, y documentar bien cuál es la utilidad de cada parámetro y cuál es el comportamiento por defecto si se omite algún parámetro opcional. Esto conforma comúnmente el manual del usuario final de la aplicación.
- Incorporar diagramas de flujo y explicaciones a nivel método de la solución, debe explicarse la estrategia general de resolución donde se pueda apreciar cómo interactúan los módulos entre sí.
- Los tipos de datos abstractos (TDAs) deben estar adecuadamente documentados en el código, por otra parte, en el manual deben constar las limitaciones posee la representación, cómo se representa una determinada estructura y detalle de métodos que provee el TDA para la manipulación de los datos.
- Incluir una sección de “Conclusiones”, donde se deben resumir complicaciones encontradas durante el desarrollo del proyecto, políticas adoptadas para su resolución, restricciones al problema original, casos particulares y finalmente aspectos relacionados a la experiencia obtenida en base a la temática del proyecto.

## Herramientas

Existen aplicaciones que permiten la generación automática de documentación para código en lenguaje C, entre las mismas se puede citar el programa **Doxygen** *http://www.doxygen.org*.