



Dogram Code

Accede a la Biblioteca Online +300 libros en PDF Gratis!

<https://dogramcode.com/biblioteca>

Únete al Canal de Telegram

https://t.me/bibliotecagratis_dogramcode

Únete al grupo de Facebook

<https://www.facebook.com/groups/librosyrecursosdeprogramacion>

Medición de Software



Mario G. Piattini Velthuis

Dogram Code

Félix O. García Rubio

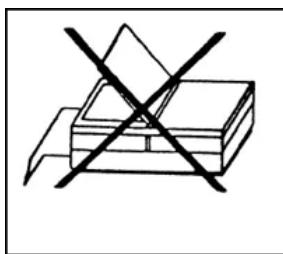


Ra-Ma®

Medición de Software

Mario G. Piattini Velthuis

Félix O. García Rubio



La ley prohíbe

fotocopiar este libro

Medición de software

© Mario G. Piattini Velthuis, Félix O. García Rubio

© De la edición: Ra-Ma 2019

MARCAS COMERCIALES. Las designaciones utilizadas por las empresas para distinguir sus productos (hardware, software, sistemas operativos, etc.) suelen ser marcas registradas. RA-MA ha intentado a lo largo de este libro distinguir las marcas comerciales de los términos descriptivos, siguiendo el estilo que utiliza el fabricante, sin intención de infringir la marca y solo en beneficio del propietario de la misma. Los datos de los ejemplos y pantallas son ficticios a no ser que se especifique lo contrario.

RA-MA es marca comercial registrada.

Se ha puesto el máximo empeño en ofrecer al lector una información completa y precisa. Sin embargo, RA-MA Editorial no asume ninguna responsabilidad derivada de su uso ni tampoco de cualquier violación de patentes ni otros derechos de terceras partes que pudieran ocurrir. Esta publicación tiene por objeto proporcionar unos conocimientos precisos y acreditados sobre el tema tratado. Su venta no supone para el editor ninguna forma de asistencia legal, administrativa o de ningún otro tipo. En caso de precisarse asesoría legal u otra forma de ayuda experta, deben buscarse los servicios de un profesional competente.

Reservados todos los derechos de publicación en cualquier idioma.

Según lo dispuesto en el Código Penal vigente, ninguna parte de este libro puede ser reproducida, grabada en sistema de almacenamiento o transmitida en forma alguna ni por cualquier procedimiento, ya

sea electrónico, mecánico, reprográfico, magnético o cualquier otro sin autorización previa y por escrito de RA-MA; su contenido está protegido por la ley vigente, que establece penas de prisión y/o multas a quienes, intencionadamente, reprodujeren o plagiaren, en todo o en parte, una obra literaria, artística o científica.

Editado por:

RA-MA Editorial

Calle Jarama, 3A, Polígono Industrial Igarsa
28860 PARACUELLOS DE JARAMA, Madrid

Teléfono: 91 658 42 80

Fax: 91 662 81 39

Correo electrónico: editorial@ra-ma.com

Internet: www.ra-ma.es y www.ra-ma.com

ISBN: 978-84-9964-850-7

Depósito legal: M-28779-2019

Maquetación: Antonio García Tomé

Diseño de portada: Antonio García Tomé

Filmación e impresión: Safekat

Impreso en España en octubre de 2019

A Javier Verdugo y Jesús Oviedo, con gratitud por su apoyo incondicional a lo largo de todos estos años.

Mario Piattini

A mis compañeros del Green Team Alarcos (@GreenTAlarcos) por su gran labor en el cuidado del medio ambiente a través de la medición para mejorar la eficiencia energética del software.

Félix García

ÍNDICE

AUTORES 11

PRÓLOGO 13

PREFACIO 17

CONTENIDO 18

ORIENTACIÓN A LOS LECTORES 19

OTRAS OBRAS RELACIONADAS 19

AGRADECIMIENTOS 21

1. INTRODUCCIÓN A LA MEDICIÓN 23

1. NECESIDAD DE MEDIR 23
2. HISTORIA DE LA MEDICIÓN 24
 1. Años 60 y 70 24
 2. Años 80 25
 3. Años 90 26
 4. Años 2000 27
 5. Años 2010 27
3. ONTOLOGIA DE LA MEDICIÓN 28
 1. Otros marcos conceptuales 33
4. LECTURAS RECOMENDADAS 37

2. GQM: GOAL QUESTION METRIC 39

1. VISIÓN GENERAL 39
 1. Fases de GQM 41
 2. Extensiones a GQM 49
 3. Casos de Aplicación de GQM 63
2. LECTURAS RECOMENDADAS 69
3. SITIOS WEB RECOMENDADOS 69

3. ESTÁNDARES DE MEDICIÓN 71

1. INTRODUCCIÓN 71
2. PRACTICAL SOFTWARE MEASUREMENT (PSM) 71
 1. Descripción 71
 2. Casos de Aplicación de PSM 73
3. ISO/IEC/IEEE 15939-75
 1. Descripción 75
 2. Casos de aplicación de ISO/IEC/IEEE 15939-78
4. LECTURAS RECOMENDADAS 80
5. SITIOS WEB RECOMENDADOS 80

4. MÉTRICAS DE SOFTWARE 81

1. INTRODUCCIÓN 81
2. MEDICIÓN DEL PROCESO 82

3. <u>MEDICIÓN DEL PROYECTO</u>	83
1. <u>Mediciones genéricas</u>	83
2. <u>Mediciones en proyectos ágiles</u>	85
3. <u>Mediciones en proyectos DevOps</u>	89
4. <u>MEDICIÓN DEL PRODUCTO</u>	100
1. <u>Métricas “Clásicas”</u>	100
2. <u>Métricas para sistemas orientados a objetos</u>	105
3. <u>Métricas MOOSE</u>	105
5. <u>LECTURAS RECOMENDADAS</u>	113
5. CONTROL ESTADÍSTICO DE PROCESOS	115
1. <u>FUNDAMENTOS DE SPC</u>	115
1. <u>Proceso de aplicación de SPC</u>	117
2. <u>Gráficos de control en SPC</u>	118
2. <u>PROCESOS SOFTWARE VS. PROCESOS INDUSTRIALES</u>	129
3. <u>CASOS DE APLICACIÓN DE SPC EN PROCESOS SOFTWARE</u>	132
1. <u>SPC: Proceso de Gestión del Proyecto</u>	132
2. <u>SPC: Proceso de Inspecciones Software</u>	135
3. <u>SPC: Proceso de Pruebas</u>	139.
4. <u>LECTURAS RECOMENDADAS</u>	142
6. IMPLANTACIÓN DE LA MEDICIÓN	143
1. <u>INTRODUCCIÓN</u>	143
2. <u>MEDICIÓN EN LOS ESTÁNDARES INTERNACIONALES</u>	143
1. <u>ISO/IEC/IEEE 12207</u>	143
2. <u>Medición en CMMI V2.0</u>	144.
3. <u>PRINCIPALES DESAFÍOS</u>	148
4. <u>FACTORES CRÍTICOS DE ÉXITO</u>	149
5. <u>CONSEJOS PRÁCTICOS</u>	152
1. <u>Consejos de gestión</u>	152
2. <u>Consejos técnicos</u>	154.
6. <u>MÉTODO DE IMPLANTACIÓN DEL PROGRAMA DE MEDICIÓN</u>	
	156
1. <u>Roles</u>	157
2. <u>Metodología</u>	157

7. LECTURAS RECOMENDADAS 159

ANEXO A. MÉTODO PARA LA CREACIÓN Y VALIDACIÓN DE MÉTRICAS 161

A.1 MÉTODO DE TRABAJO 162

A.2 IDENTIFICACIÓN 163

A.3 CREACIÓN 164

A.3.1 Definición 165

A.3.2 Validación teórica 165

A.3.3 Validación empírica 166

A.3.4 Explicación psicológica 166

A.4 ACEPTACIÓN 166

A.5 APLICACIÓN 166

A.6 ACREDITACIÓN 167

A.7 LECTURAS RECOMENDADAS 167

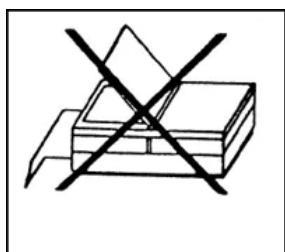
ACRÓNIMOS 169

REFERENCIAS 173

Medición de Software

Mario G. Piattini Velthuis

Félix O. García Rubio



Medición de software

© Mario G. Piattini Velthuis, Félix O. García Rubio

© De la edición: Ra-Ma 2019

MARCAS COMERCIALES. Las designaciones utilizadas por las empresas para distinguir sus productos (hardware, software, sistemas operativos, etc.) suelen ser marcas registradas. RA-MA ha intentado a lo largo de este libro distinguir las marcas comerciales de los términos descriptivos, siguiendo el estilo que utiliza el fabricante, sin intención de infringir la marca y solo en beneficio del propietario de la misma. Los datos de los ejemplos y pantallas son ficticios a no ser que se especifique lo contrario.

RA-MA es marca comercial registrada.

Se ha puesto el máximo empeño en ofrecer al lector una información completa y precisa. Sin embargo, RA-MA Editorial no asume ninguna responsabilidad derivada de su uso ni tampoco de cualquier violación de patentes ni otros derechos de terceras partes que pudieran ocurrir. Esta publicación tiene por objeto proporcionar unos conocimientos precisos y acreditados sobre el tema tratado. Su venta no supone para el editor ninguna forma de asistencia legal, administrativa o de ningún otro tipo. En caso de precisarse asesoría legal u otra forma de ayuda experta, deben buscarse los servicios de un profesional competente.

Reservados todos los derechos de publicación en cualquier idioma.

Según lo dispuesto en el Código Penal vigente, ninguna parte de este libro puede ser reproducida, grabada en sistema de almacenamiento o transmitida en forma alguna ni por cualquier procedimiento, ya sea electrónico, mecánico, reprográfico, magnético o cualquier otro sin autorización previa y por escrito de RA-MA; su contenido está protegido por la ley vigente, que establece penas de prisión y/o multas a quienes, intencionadamente, reprodujeren o plagiaren, en todo o en parte, una obra literaria, artística o científica.

Editado por:

RA-MA Editorial

Calle Jarama, 3A, Polígono Industrial Igarsa
28860 PARACUELLOS DE JARAMA, Madrid

Teléfono: 91 658 42 80

Fax: 91 662 81 39

Correo electrónico: editorial@ra-ma.com

Internet: www.ra-ma.es y www.ra-ma.com

ISBN: 978-84-9964-850-7

Depósito legal: M-28779-2019

Maquetación: Antonio García Tomé

Diseño de portada: Antonio García Tomé

Filmación e impresión: Safekat

Impreso en España en octubre de 2019

A Javier Verdugo y Jesús Oviedo, con gratitud por su apoyo incondicional a lo largo de todos estos años.

Mario Piattini

A mis compañeros del Green Team Alarcos (@GreenTAlarcos) por su gran labor en el cuidado del medio ambiente a través de la medición para mejorar la eficiencia energética del software.

Félix García

ÍNDICE

[AUTORES 11](#)

[PRÓLOGO 13](#)

[PREFACIO 17](#)

[CONTENIDO 18](#)

[ORIENTACIÓN A LOS LECTORES 19](#)

[OTRAS OBRAS RELACIONADAS 19](#)

[AGRADECIMIENTOS 21](#)

1. INTRODUCCIÓN A LA MEDICIÓN 23

1. [NECESIDAD DE MEDIR 23](#)

2. [HISTORIA DE LA MEDICIÓN 24](#)

1. [Años 60 y 70 24](#)

2. [Años 80 25](#)

3. [Años 90 26](#)

4. [Años 2000 27](#)

5. [Años 2010 27](#)

3. [ONTOLOGIA DE LA MEDICIÓN 28](#)

1. [Otros marcos conceptuales 33](#)

4. [LECTURAS RECOMENDADAS 37](#)

2. GQM: GOAL QUESTION METRIC 39

1. [VISIÓN GENERAL 39](#)

1. [Fases de GQM 41](#)

2. [Extensiones a GQM 49](#)

3. [Casos de Aplicación de GQM](#) 63
2. [LECTURAS RECOMENDADAS](#) 69.
3. [SITIOS WEB RECOMENDADOS](#) 69.

3. ESTÁNDARES DE MEDICIÓN 71

1. [INTRODUCCIÓN](#) 71
2. [PRACTICAL SOFTWARE MEASUREMENT \(PSM\)](#) 71
 1. [Descripción](#) 71
 2. [Casos de Aplicación de PSM](#) 73
3. [ISO/IEC/IEEE 15939-75](#)
 1. [Descripción](#) 75
 2. [Casos de aplicación de ISO/IEC/IEEE 15939](#) 78
4. [LECTURAS RECOMENDADAS](#) 80
5. [SITIOS WEB RECOMENDADOS](#) 80

4. MÉTRICAS DE SOFTWARE 81

1. [INTRODUCCIÓN](#) 81
2. [MEDICIÓN DEL PROCESO](#) 82
3. [MEDICIÓN DEL PROYECTO](#) 83
 1. [Mediciones genéricas](#) 83
 2. [Mediciones en proyectos ágiles](#) 85
 3. [Mediciones en proyectos DevOps](#) 89
4. [MEDICIÓN DEL PRODUCTO](#) 100
 1. [Métricas “Clásicas”](#) 100
 2. [Métricas para sistemas orientados a objetos](#) 105
 3. [Métricas MOOSE](#) 105
5. [LECTURAS RECOMENDADAS](#) 113

5. CONTROL ESTADÍSTICO DE PROCESOS 115

1. [FUNDAMENTOS DE SPC](#) 115
 1. [Proceso de aplicación de SPC](#) 117
 2. [Gráficos de control en SPC](#) 118
2. [PROCESOS SOFTWARE VS. PROCESOS INDUSTRIALES](#) 129
3. [CASOS DE APLICACIÓN DE SPC EN PROCESOS SOFTWARE](#) 132
 1. [SPC: Proceso de Gestión del Proyecto](#) 132
 2. [SPC: Proceso de Inspecciones Software](#) 135

- 3. SPC: Proceso de Pruebas 139
 - 4. LECTURAS RECOMENDADAS 142
- 6. IMPLANTACIÓN DE LA MEDICIÓN 143**
- 1. INTRODUCCIÓN 143
 - 2. MEDICIÓN EN LOS ESTÁNDARES INTERNACIONALES 143
 - 1. ISO/IEC/IEEE 12207 143
 - 2. Medición en CMMI V2.0 144
 - 3. PRINCIPALES DESAFÍOS 148
 - 4. FACTORES CRÍTICOS DE ÉXITO 149
 - 5. CONSEJOS PRÁCTICOS 152
 - 1. Consejos de gestión 152
 - 2. Consejos técnicos 154
- 6. MÉTODO DE IMPLANTACIÓN DEL PROGRAMA DE MEDICIÓN 156**
- 1. Roles 157
 - 2. Metodología 157
7. LECTURAS RECOMENDADAS 159

ANEXO A. MÉTODO PARA LA CREACIÓN Y VALIDACIÓN DE MÉTRICAS 161

- A.1 MÉTODO DE TRABAJO 162
- A.2 IDENTIFICACIÓN 163
- A.3 CREACIÓN 164
 - A.3.1 Definición 165
 - A.3.2 Validación teórica 165
 - A.3.3 Validación empírica 166
 - A.3.4 Explicación psicológica 166
- A.4 ACEPTACIÓN 166
- A.5 APLICACIÓN 166
- A.6 ACREDITACIÓN 167
- A.7 LECTURAS RECOMENDADAS 167

ACRÓNIMOS 169

REFERENCIAS 173

AUTORES

MARIO GERARDO PIATTINI VELTHUIS

Doctor y Licenciado en Informática por la Universidad Politécnica de Madrid. Licenciado en Psicología por la Universidad Nacional de Educación a Distancia. Máster en Auditoría Informática (CENEI), Máster en Dirección de RR.HH. (IMAFE) y Master´s Certificate en Dirección de Proyectos (George Washington University). Especialista en la Aplicación de Tecnologías de la Información en la Gestión Empresarial (CEPADE-UPM). CISA (Certified Information System Auditor), CISM (Certified Information System Manager), CRISC (Certified in Risk and Information System Control) y CGEIT (Certified in the Governance of Enterprise IT) por la ISACA. PMP (Project Management Professional) por el PMI. Diplomado en Calidad por la Asociación Española para la Calidad. Auditor Jefe ISO 15504/33000 por AENOR.

Ha trabajado como consultor para numerosos organismos y empresas, entre los que destacan: Ministerio de Industria y Energía, Ministerio de Administraciones Públicas, Siemens-Nixdorf, Unisys, Hewlett-Packard, Oracle, ICM, Atos-Ods, Avanzit, Sistemas Técnicos de Loterías, Indra/Soluziona, Alhambra/Eidos, Mundo Reader (BQ), etc. Socio fundador de las empresas Cronos Ibérica S.A. (actualmente Alten), Kybele Consulting S.L. (actualmente Intelligent Environments), Lucentia Lab, S.L., DQTeam, S.L. y AQCLab, primer laboratorio acreditado por ENAC para la evaluación de la calidad de producto software y de los datos. Ha sido profesor asociado en la Universidad Complutense y en la Universidad Carlos III de Madrid. Ha sido Director del Centro Mixto de Investigación y Desarrollo de Software UCLM-Indra, Coordinador del Área de Ciencias de la Computación y Tecnología Informática de la Agencia Nacional de Evaluación y Prospectiva (ANEPE), y Director del Instituto de Tecnologías y Sistemas de Información (ITSI) de la UCLM. <https://dogramcode.com/programacion>

Catedrático de Universidad de Lenguajes y Sistemas Informáticos en la Escuela Superior de Informática (ESI) de la Universidad de Castilla-La Mancha (UCLM), donde dirige el grupo de investigación Alarcos, especializado en Calidad de Sistemas de Información.

Entre los 15 “*Top scholars in the field of systems and software engineering (2004-2008)*” y entre los 15 “*Most active experienced SE researchers (2010-2017)*”, Premio Nacional a la Trayectoria Profesional del Ingeniero Informático de la Federación de Asociaciones de Ingenieros Informáticos de España, y Premio Aritmel por la Sociedad Científica Informática de España (SCIE).

FÉLIX ÓSCAR GARCÍA RUBIO

Doctor por la Universidad de Castilla-La Mancha, en la que también obtuvo los títulos de Ingeniero en Informática e Ingeniero Técnico en Informática de Gestión. Catedrático de Universidad en la Escuela Superior de Informática de Ciudad Real. Es miembro del grupo de investigación Alarcos especializado en sistemas de información, bases de datos e ingeniería del software. Sus temas de investigación incluyen la calidad de los procesos software, la medición, los métodos ágiles, los procesos de negocio y la sostenibilidad del software. Sobre estos temas ha escrito varios libros, capítulos de libro y diversos artículos en revistas y conferencias nacionales e internacionales. Certificado PMP (Project Management Professional), CISA (Certified Information Systems Auditor) y Scrum Manager (Nivel Experto).

PRÓLOGO

La importancia del software en nuestra sociedad es evidente. De hecho, la necesidad de construir aplicaciones software de forma correcta, fiable y predecible se hizo patente desde la mitad del siglo pasado, y fue recogida de forma explícita en la conferencia de la NATO de 1968, que acuñó por

primera vez el término “ingeniería de software” y cuya definición fue consensuada un poco más tarde por las principales asociaciones y organizaciones internacionales, como ISO, IEC o IEEE:

La ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; es decir, la aplicación de la ingeniería al software.

Si analizamos la definición, lo primero que hace es dejar claro que el desarrollo de software no es la única actividad que debe preocupar a un ingeniero de software, sino que la operación y el mantenimiento son también fundamentales. En concreto, se estima que las labores de desarrollo suelen ocupar algo menos de un 30 por ciento de la vida de cualquier aplicación software, estando el restante 70 por ciento de la vida útil de un programa dedicado a labores de su mantenimiento y evolución.

Por otro lado, la primera parte de la definición de ingeniería de software establece los tres pilares fundamentales de cualquier ingeniería: el uso de procesos **sistemáticos** basados en principios y metodologías científicamente probados y bien asentados; un enfoque **disciplinado** que permita la reproducibilidad de los productos y resultados obtenidos; y la capacidad de poder medir de forma objetiva y **cuantificable** tanto los procesos utilizados como los productos obtenidos, así como la calidad de los mismos.

Este libro se centra precisamente en uno de esos tres pilares fundamentales: la *medición del software*. Aunque de apariencia sencilla, en el caso del software el tema de la medición se torna muy complejo entre otras cosas por una de las características que distingue al software de los artefactos de otras ingenierías más tradicionales: su intangibilidad. Somos capaces de contar el número de líneas de un código de un programa, el número de clases de un modelo de software, o las llamadas que se producen a un método concreto. Pero ¿cómo cuantificar la usabilidad de una aplicación o su mantenibilidad?

¿cómo estimar el tiempo y los costes de su desarrollo? ¿cómo definir indicadores de su calidad?

El tema de la medición ha preocupado mucho a la comunidad que se dedica a la ingeniería de software, puesto que sin ser capaces de medir los artefactos resultantes (productos, aplicaciones, servicios software) o los procesos que guían su desarrollo, operación y mantenimiento, no es posible evaluar su calidad, controlarlos o mejorarlos. Así, las principales asociaciones relacionadas con la ingeniería de software (como ACM e IEEE) y los organismos de normalización internacionales (como ISO, OMG o IEC) han estado trabajando en los últimos 30 años en estos temas, para los que se han desarrollado diversas normas y recomendaciones internacionales.

Igualmente, la industria del software, muy consciente también de la necesidad de estimar el coste de sus desarrollos y evaluar de forma precisa la calidad de los productos resultantes, ha acumulado una gran experiencia en estos temas. Porque una cosa es la definición teórica de las medidas y los procesos de medición, y otra su implantación de forma práctica, realista y efectiva. Una de las primeras cosas que se aprende cuando se está a cargo de un proyecto de software de cierta complejidad es a definir las medidas significativas y apropiadas en cada caso, y a integrar en el proyecto los procesos de medición y evaluación de su calidad desde el primer momento, de una forma lo menos invasiva y lo más automatizada posible. Cualquier otra estrategia lleva a que los procesos de medición se conviertan en un problema en vez de en una ayuda, y a que las medidas resultantes dejen de ser efectivas, o incluso fiables; es decir, podemos terminar complicando los procesos de producción y ayudando a incrementar los costes finales, en vez de a reducirlos. De ahí la necesidad de implantar medidas y procesos apropiados de medición del software, y la importancia de conocerlos.

En este sentido, este libro llena un vacío importante en la bibliografía sobre temas de ingeniería de software, puesto que proporciona los conceptos básicos de la medición, así como las normas internacionales de aplicación en

estos temas, ofreciendo una visión completa y muy actualizada sobre sus principales técnicas y fundamentos; pero lo hace sin perder de vista la necesidad de tener que implantar dichos conceptos en entornos de producción de aplicaciones, aportando un conjunto de recomendaciones, guías y experiencias muy útiles para cualquier profesional que haya de poner esos conceptos en práctica, conjugando perfectamente el rigor científico con el pragmatismo necesario en cualquier industria.

Es para mí un honor poder escribir las primeras líneas de un libro como este, y espero que disfruten y aprendan de su lectura tanto como yo lo he hecho. Personalmente creo que los autores han realizado una labor excelente ofreciéndonos una obra de referencia para todos aquellos estudiantes y profesionales que necesiten conocer los conceptos y técnicas fundamentales de la medición de software.

No subestimen la importancia de la medición del software en la construcción y mantenimiento de aplicaciones como las que demanda actualmente la sociedad. Hoy en día, más que nunca, el software necesita de la ingeniería, y en particular de procesos de medición que sean útiles y efectivos, que permitan evaluar su calidad y ayudar a mejorarla.

Antonio Vallecillo

Catedrático de Lenguajes y Sistemas Informáticos de la Universidad de Málaga

Miembro del subcomité SC7 de ISO, Ingeniería de Software y de Sistemas
Expresidente de la Sociedad Nacional de Ingeniería de Software

PREFACIO

La medición nos acompaña desde que nacemos (test de Apgar¹) hasta que dejamos este mundo (ya que para declarar la muerte clínica y la biológica también se miden varios indicadores). En nuestra vida, continuamente

realizamos o utilizamos mediciones que nos guían a la hora de tomar decisiones y seleccionar la alternativa que creemos mejor, como por ejemplo a la hora de comprar un producto, seleccionar la ruta a seguir en nuestras vacaciones, decidir ir al médico si tenemos fiebre, etc.

Por supuesto, la medición es fundamental en los aspectos relativos a la calidad. De hecho, tanto la gestión de calidad total, como la familia de normas ISO 9000 o modelos como Seis Sigma insisten en la propuesta de objetivos cuantificados, el benchmarking y la toma de decisiones basadas en hechos.

También en la gestión de proyectos -PMBOK (Project Management Body of Knowledge) del PMI (Project Management Institute)- o en la de servicios -en modelos como VeriSM (Value-Driven Evolving, Responsive Integrated Service Management), ITIL, la familia de norma ISO/IEC 20000, o CMMI-SVC (CMMI Services)-, la definición de indicadores y métricas resulta uno de los aspectos claves.

Y por supuesto todos los modelos de calidad software relacionados con las personas -People CMM, PSP (Personal Software Process), TSP (Team Software Process), CMMI-PPL (CMMI People Management), etc.-, los procesos (las diferentes versiones de CMMI Development, CMMI Supplier Management, ISO/IEC 29110, ISO/IEC 33000, etc.), y los productos (ISO/IEC 25000) también destacan la importancia de disponer de métricas útiles bien definidas y validadas.

La presente obra reúne diferentes aspectos sobre la medición, que pueden ser aplicados tanto a los proyectos, servicios, procesos o productos software. Para lo cual se persiguen los siguientes objetivos:

- Presentar de forma clara los conceptos fundamentales relacionados con la medición.
- Exponer los aspectos más significativos relacionados con la medición de productos, procesos y proyectos software.

- Dar a conocer los diferentes estándares relacionados con este tema.
- Analizar el importante papel que juega la medición en el aseguramiento y control de la calidad del software.

A lo largo de esta obra se ha combinado el rigor científico con la experiencia práctica, proporcionando una panorámica actual y completa sobre la problemática asociada a la calidad de los sistemas de medición del software.

CONTENIDO

El capítulo 1 presenta una perspectiva de la medición del software a lo largo de varias décadas, así como los conceptos básicos relacionados con la medición (utilizando para ello una ontología), presentes en varios estándares internacionales.

El siguiente capítulo se centra en la metodología GQM, sin duda, una de las más importantes para desarrollar un conjunto de métricas de manera rigurosa y que respondan a objetivos concretos. También se exponen las diferentes extensiones, así como los casos de aplicación llevados a cabo en los últimos años.

En el capítulo 3 se resume PSM, que dio lugar al estándar más importante sobre el proceso de medición: el ISO/IEC/IEEE 15939, que se describe junto con las aplicaciones que ha tenido en el campo del software.

En el capítulo 4, se presenta una colección de métricas de software que aborda las distintas vistas de la medición del software: el proceso, el proyecto y el producto.

El control estadístico de procesos, y especialmente su aplicación en el campo de los procesos software, se aborda en el capítulo 5.

El capítulo 6 se dedica a abordar el difícil reto de la implantación de programas de medición en las organizaciones de desarrollo de software.

Se incluye también un anexo que presenta un método para la creación y validación de métricas. Por último, se encuentran los acrónimos y la bibliografía utilizados en el texto.

ORIENTACIÓN A LOS LECTORES

Aunque un conocimiento en profundidad de la medición del software y su calidad puede estar reservado a expertos en la materia, nuestro propósito al realizar este libro ha sido dirigirnos a una audiencia mucho más amplia que comprende:

- Alumnos de grado y postgrado de Informática, Sistemas de Información, Ingeniería del Software, etc.
- Participantes en seminarios o cursos monográficos sobre calidad de sistemas de información, calidad de software, métricas, etc.
- Profesionales informáticos que estén trabajando en el área del desarrollo de sistemas de información o en aseguramiento de calidad.
- Directivos que tengan entre sus responsabilidades el gobierno o la calidad de los sistemas de información.
- Usuarios avanzados, que tengan interés en adquirir unos conocimientos sobre las métricas más utilizadas para asegurar la calidad del software.
- Analistas o consultores que, aun teniendo conocimientos de la materia, quieran abordarla de forma más sistemática.

Debido a la diversidad de la audiencia, el estudio de esta obra puede realizarse de maneras muy distintas dependiendo de la finalidad y conocimientos previos del lector.

OTRAS OBRAS RELACIONADAS

Queremos destacar que existen algunos libros que complementan la visión de la presente obra:

- *Calidad de Sistemas de Información, 5^a edición, Piattini, M., García, F. Pino, F. y García, I., 2019, Ra-Ma, Madrid*, que aporta una visión general sobre las principales técnicas para la calidad de los procesos, proyectos, y personas relacionados con los sistemas software.
- *Calidad del Software. Evaluación y Certificación de Productos Software. Rodríguez, M. y Piattini, M., 2019, Madrid, Ra-Ma*, que profundiza en los modelos de calidad y métricas para el producto software.
- *Calidad de Datos. Caballero, I., Gómez, A., Merino, J. y Piattini, M., 2018, Madrid, Ra-Ma*, que ofrece un tratamiento sistemático de las principales técnicas para el gobierno, la gestión y la calidad de los datos y su medición.
- *Mantenimiento y Evolución de Sistemas de Información. Pérez, R., García, I., Ruiz, F., Polo, M. y Piattini, M., 2018. Ra-Ma, Madrid*, que presenta las principales técnicas de mantenimiento (reingeniería, modernización, etc.) que permiten mejorar la calidad de los sistemas de información y su evolución, así como las métricas relacionadas.
- *Métodos de Investigación en Ingeniería del Software. 2^a ed. Genero, M., Cruz-Lemus, J.A., Díaz, O. y Piattini, M., 2020, Madrid, Ra-Ma*, que ofrece una visión general de la utilización de las encuestas, experimentos, estudios de caso e investigación en acción en la Ingeniería del Software, que permiten validar métricas de manera empírica.

Y otros profundizan en el gobierno o calidad de los sistemas de información:

- *Calidad de Servicios basados en Tecnologías de la Información.* Piattini, M., 2019, Madrid, Ra-Ma, que analiza los principales conceptos y elementos relacionados con la calidad de los servicios y particularmente de aquellos servicios soportados por las Tecnologías de la Información.
- *Gobierno y Gestión de las Tecnologías y los Sistemas de Información basados en Arquitectura Empresarial.* Piattini, M. y Ruiz, F., 2020, Madrid, Ra-Ma, en el que se tratan estándares y metodologías para llevar a cabo el gobierno y gestión de los sistemas de información, optimizando los beneficios y minimizando los riesgos.
- *Gobierno de Datos.* Caballero, I. y Piattini, M., 2020, Madrid, Ra-Ma, en el que se tratan los principales modelos de gobierno de datos, así como los componentes de un sistema de gobierno de datos.

AGRADECIMIENTOS

Querríamos expresar nuestro agradecimiento, en primer lugar, a los alumnos de las asignaturas *Calidad de Sistemas de Información*, *Calidad del Software y Gestión*, *Certificación y Evaluación de Sistemas y Servicios* de la Escuela Superior de Informática de Ciudad Real, así como a los asistentes a los diferentes seminarios y conferencias que hemos organizado sobre diferentes aspectos de la calidad de los sistemas informáticos y específicamente en la medición del software en los más de veinticinco años que llevamos trabajando sobre el tema. También a los compañeros de AQCLab que batallan con los problemas de calidad en el día a día.

También deseamos dar las gracias a D. Antonio Valleclillo, por haber aceptado escribir la presentación de esta obra. Por último, nos resta dar las gracias a Sandra Ramírez por sus valiosas sugerencias que, como en otras muchas ocasiones, han contribuido a mejorar considerablemente este libro, y a la editorial Ra-Ma, especialmente a Jesús Ramírez Martínez y Jesús Ramírez Galán, por su apoyo y confianza.

Mario G. Piattini Velthuis

Félix Ó. García Rubio

Ciudad Real, septiembre 2019.

1 https://es.wikipedia.org/wiki/Test_de_Apgar

<https://dogramcode.com/programacion>

1

INTRODUCCIÓN A LA MEDICIÓN

NECESIDAD DE MEDIR

No le faltaba razón al físico Lord Kelvin cuando a finales del siglo XIX afirmaba que: “*cuando puedes medir sobre lo que estás diciendo, y expresarlo en números, sabes algo sobre ello, pero cuando no puedes medirlo ni expresarlo en números entonces tu conocimiento es precario e insatisfactorio*”. De hecho, la medición es fundamental en nuestra vida cotidiana. Continuamente realizamos o utilizamos mediciones que nos guían a la hora de tomar decisiones y seleccionar la alternativa que creemos mejor, como por ejemplo a la hora de comprar un producto, seleccionar la ruta a seguir en nuestras vacaciones, decidir ir al médico si tenemos fiebre, etc.

La medición cuenta con una larga tradición y constituye una disciplina fundamental en cualquier ingeniería, y la ingeniería del software no es una excepción. La medición software es una disciplina joven, no obstante, está

en constante crecimiento y la importancia que ha adquirido es cada vez más evidente.

La necesidad y motivación por medir está ligada fundamentalmente a la mejora y de hecho una de las razones principales del incremento masivo en el interés en las métricas ha sido la percepción de que son necesarias para la mejora de la calidad del software (Fenton, 2001). En efecto, para poder asegurar que un proceso o sus productos resultantes son de calidad o poder compararlos, se deben asignar valores, descriptores, indicadores o algún otro mecanismo mediante los cuales se pueda llevar a cabo dicha comparación. Para ello, es necesario llevar a cabo un proceso de medición del software, que en general, persigue tres objetivos fundamentales: ayudarnos a entender qué ocurre durante el desarrollo y el mantenimiento, permitirnos controlar qué es lo que ocurre en los proyectos, y poder mejorar los procesos y productos (Fenton y Bieman 2014). En efecto, las métricas son un buen medio para entender, monitorizar, controlar, predecir y probar el desarrollo de software y los proyectos de mantenimiento (Briand et al., 1996a) y pueden ser utilizadas por profesionales e investigadores para tomar mejores decisiones (Pfleeger, 1997).

HISTORIA DE LA MEDICIÓN

Los orígenes de la medición del software se sitúan en torno a finales de los años sesenta, en los que aparecieron propuestas sobre los que se asentaron las bases para una mayor intensidad de trabajo en el área hasta la actualidad. Las principales etapas que se pueden distinguir en esta evolución son las siguientes (Ott, 1996; Zuse 1998):

Años 60 y 70

El trabajo más antiguo de definición de métricas data del año 1968, en el que Rubey y Hartwick (1968) presentaron una métrica para la definición de la calidad del software, basada en el número de instrucciones originales y

modificadas durante la ejecución, así como los comentarios que incluyen las instrucciones del programa. Previamente a este trabajo se habían propuesto algunos modelos de estimación, como el de Delphi (Helmer-Heidelberg, 1966) y SDC de (Nelson, 1966).

En este periodo, en el que surge el paradigma estructurado (Yourdon, De Marco, etc.), las principales razones para concebir métricas software se basan en la influencia que la estructura del programa tiene sobre la fiabilidad del software, persiguiéndose mantener un alto grado de modularidad. En este sentido, Parnas (1975) establece que los módulos se deberían reestructurar de modo que no tengan conocimiento de la estructura interna de otros módulos. Por ello, la medición del software se centró en la evaluación de la complejidad del software, siendo las dos métricas más destacadas en este periodo la famosa **métrica de complejidad ciclomática** (McCabe, 1976) y la **métrica de la “ciencia del software”** de Halstead (1977). A finales de los 70 se produjo una gran proliferación de métricas de complejidad, basadas en la idea de mejorar las métricas propuestas previamente, alcanzándose un estado en el que Curtis (1979) llegó a afirmar que “*hay más métricas de complejidad que informáticos en activo*”.

Otra de las métricas que merece especial atención en este periodo es la métrica **Líneas de Código Fuente** (LOC, *Lines of Code*). Wolverton (1974) hizo uno de los primeros intentos para definir formalmente la productividad de un programador usando LOC. Se trata de una de las métricas que ha estado sujeta a mayor debate y críticas, a pesar o debido a su aparente simplicidad, y ha habido numerosas interpretaciones de la misma, que han dado lugar a miles de artículos.

De los finales de los 70 también hay una medida de estimación que merece especial atención, los **puntos función**, definidos por Albrecht (1979), y que sigue siendo hoy en día una métrica fundamental para la gestión de

proyectos, que se aplica ampliamente con las lógicas adaptaciones a la propuesta original.

A finales de este periodo también se propusieron las primeras métricas aplicables a la fase de diseño, en trabajos como el de Gilb (1977) -que además constituyó uno de los primeros libros sobre medición software-, Yin y Winchester (1978) y Chapin (1979). Del mismo modo, a finales de los 70 empieza a haber cierta repercusión del campo de la medición del software en los foros científicos de discusión, como se demuestra con la aparición de talleres especializados como el *IEEE Workshop on Quantitative Software Models* (en 1979) o un número especial de la revista *IEEE Transactions on Software Engineering*, en el que la investigación sobre métricas cobraba un papel protagonista.

Años 80

En los inicios de los años 80 se mantenía un cierto escepticismo sobre la aportación del campo de la medición del software, debido a una serie de críticas que se centraban en métricas específicas, como las críticas realizadas sobre las imprecisiones de la métrica de Halstead (Coulter, 1983; Lassez et al., 1981) o la propia métrica de McCabe (Evangelist, 1984); o críticas en general sobre la precisión o simplicidad de las métricas definidas para medir atributos de calidad como la complejidad y sobre todo la falta de validación de las métricas definidas (Curtis, 1981; Belady, 1981). Estas críticas resaltaron la importancia de realizar con rigor la investigación en el campo de las métricas software, de modo que en este periodo, aparte de haber una cierta continuidad en la investigación sobre métricas de complejidad del software aplicadas al diseño, como por ejemplo se aprecia en los trabajos de (Henry y Kafura, 1981; Card y Agresti, 1988; McCabe y Butler, 1989), se comienza a trabajar en otras áreas complementarias y esenciales en el campo de la medición como son los métodos, siendo el ejemplo más conocido la propuesta **GQM (Goal Question Metric)** (Basili y Weiss, 1984), que plantea la necesidad de definir métricas siempre orientadas a

alcanzar un objetivo que se refina en forma de preguntas. Otros aspectos en los que se empieza a avanzar y que aportan un enfoque ingenieril al campo de la medición del software son la teoría de la medición (DeMillo y Lipton, 1981; Baker et al., 1987) y que se fue asentando en los 90 (Fenton y Melton, 1990; Fenton, 1991; Zuse, 1991). En relación a estos trabajos, Weyuker (1988) propuso un modelo de validación formal de las métricas software basado en propiedades.

En esta década Boehm (1981) creó el modelo COCOMO, uno de los modelos de estimación de costes más aplicado y que se basa en la métrica LOC.

También merece especial atención la investigación en técnicas estadísticas y de experimentación, con el fin de fomentar la validación de las métricas. En este sentido cabe destacar trabajos tales como el de Basili et al. (1986), que proponen un marco de trabajo para ayudar al proceso experimental en el contexto del proyecto TAME (*Tailoring a Measurement Environment*) (Basili y Rombach 1987).

A finales de los 80 también encontramos los primeros esfuerzos por la estandarización en el campo de la medición del software (IEEE, 1989). No obstante, la mayoría de las métricas abordadas en dichos informes eran de proceso centradas en la fase de mantenimiento.

Otro de los aspectos a mencionar en este periodo es la proliferación de proyectos de investigación sobre medición software tanto en Estados Unidos, encabezados mayoritariamente por el SEI (Software Engineering Institute) y la (NASA, 1981), como en Europa, en el que se desarrollan proyectos tales como METKIT (*Metrics Educational Toolkit*), COSMOS (*Cost Management with Metrics of Specification*) y MERMAID (*Metrication and Resource Modelling Aid*).

Años 90

En paralelo al auge del paradigma de orientación a objetos (OO), a finales de los 80 y fundamentalmente en los 90, se propusieron una gran cantidad de

métricas para OO (Chidamber y Kemerer 1994, Brito e Abreu y Carapuça, 1994; Lorenz y Kidd 1994), y para reutilización de software (Bieman, 1991). En el año 1993 se publicó el estándar IEEE 1061, el cual proporcionaba una guía sobre cómo validar métricas software.

Durante este periodo se asentaron las bases iniciadas en el periodo anterior, en relación a la validación teórica de las métricas, con la aparición de nuevos marcos de trabajo (Briand et al., 1996b; Whitmire, 1997; Zuse, 1998) y en validación empírica (Basili et al., 1999). También se sigue avanzando en la parte metodológica, con refinamientos al método GQM y propuestas de extensiones, como la metodología Goal Driven Measurement (Park et al., 1996). En esta década se lanza la conferencia METRICS (*International Software Metrics Symposium*), cuya primera edición se celebró en Baltimore (EEUU), patrocinada por IEEE Computer Society.

También en esta década se desarrollan las diferentes variantes de los puntos función (IFPUG, Mk II, Nesma, FiSMA FSM, COSMIC FFP, etc.)

Años 2000

En la década de los 2000, siguió la tendencia de la importancia creciente de la validación empírica de las métricas, como demuestra la proliferación de trabajos en este campo (Wohlin et al., 2000; Perry et al., 2000; Juristo y Moreno, 2001). Otro ejemplo es la aparición de la conferencia ISESE (*International Symposium on Experimental Software Engineering*) cuya primera edición se celebró en Nara (Japón) en el año 2002. Las conferencias METRICS e ISESE se fusionaron a partir del año 2007 (edición celebrada en Madrid, España) en el denominado *International Symposium on Empirical Software Engineering and Measurement* (ESEM). Actualmente se celebra de manera anual la “Empirical Software Engineering International Week” (ESEIW) en la que se presentan los principales avances en cuanto a la validación y aplicación empírica de las métricas.

También es destacable en este periodo el énfasis en la parte metodológica, surgiendo nuevos enfoques como PSM (*Practical Software & Systems Measurement*), y el importante papel que empieza a asumir la medición del software en los estándares internacionales, mediante el desarrollo de estándares específicos como la ISO/IEC/IEEE 15939, o su presencia significativa en otros estándares relacionados como ISO/IEC 25000, ISO/IEC 12207 o ISO 9000.

Durante esta década y tras la publicación del Manifiesto Ágil en 2001, se propone un conjunto de métricas ágiles tanto para la estimación como para el control de los proyectos.

Años 2010

En 2010 el Object Management Group (OMG) y el Software Engineering Institute (SEI) de la Universidad Carnegie Mellon fundan el CISQ (Consortium for IT Software Quality), que durante la década actual propone varias métricas para la calidad del software.

Por su parte, en esta década ISO publica diversos estándares para la medición de software, como el ISO/IEC 25022 y el ISO/IEC 25023, y otros como el ISO/IEC 25024 para calidad de datos y el ISO/IEC 25025 para calidad de servicios de TI.

También se dedica gran parte de la investigación a caracterizar y medir la llamada “deuda técnica” o “deuda de diseño” (Pérez et al., 2018).

Desde mediados de la década, con la difusión de la ingeniería del software continua y “DevOps”, se desarrollan varios indicadores que permiten monitorizar el estado de este tipo de proyectos.

ONTOLOGÍA DE LA MEDICIÓN

En relación a la terminología relacionada con la medición software, tal como se ha comentado en el primer apartado, la medición de software es una disciplina relativamente joven y como consecuencia no existe aún un

consenso general sobre la definición exacta de los conceptos que maneja. A pesar de contar con diversos estándares internacionales que tratan de normalizar estos temas, se han detectado ciertas lagunas e inconsistencias en los términos que dichos estándares definen, como son por ejemplo los conceptos de medida, métrica, medición, indicador, etc. La situación no es mucho mejor en los contextos académicos e industriales, en donde las distintas propuestas de modelos de medición de software tampoco han logrado consensuar una terminología coherente y ampliamente aceptada entre toda la comunidad científica.

En este apartado resumiremos los principales conceptos básicos de la medición siguiendo la “ontología de la medición” (García et al., 2005) y que se representa en la Figura 1.1.

Así, suele existir en las organizaciones una “necesidad de medir” o **necesidad de información**, por ejemplo, de conocer el nivel de productividad de los programadores de un proyecto en comparación con lo habitual en otros proyectos en la organización. Esta necesidad de medir está asociada a un **concepto medible**, que se puede definir como una “relación abstracta entre atributos y necesidades de información”; por ejemplo, el ratio de productividad de un equipo de desarrollo frente a un grado de productividad objetivo. Un **atributo** será una “propiedad mensurable, física o abstracta, que comparten todas las **entidades** de una **categoría de entidad**”, como el tamaño de un código fuente; pudiendo definir categoría de entidad como una colección de entidades caracterizadas por satisfacer un cierto predicado común; por ejemplo, los programas escritos en Python.

Una **entidad** es “un objeto que va a ser caracterizado mediante una medición de sus atributos”. En medición del software se pueden distinguir entre los siguientes tipos de entidades: *Procesos*, en el que se incluyen las mediciones relacionadas a las actividades del software; *Proyectos*, que son el resultado de la ejecución de los procesos y de los que se obtienen como

resultado *Productos*, que incluyen los entregables y documentos resultantes de las actividades y se utilizan *Recursos*, que incluye los recursos necesarios para el desarrollo de los proyectos software tales como personal, software, hardware, etc. y *Servicios*, que se proporcionan utilizando las entidades anteriores.

<https://dogramcode.com/programacion>

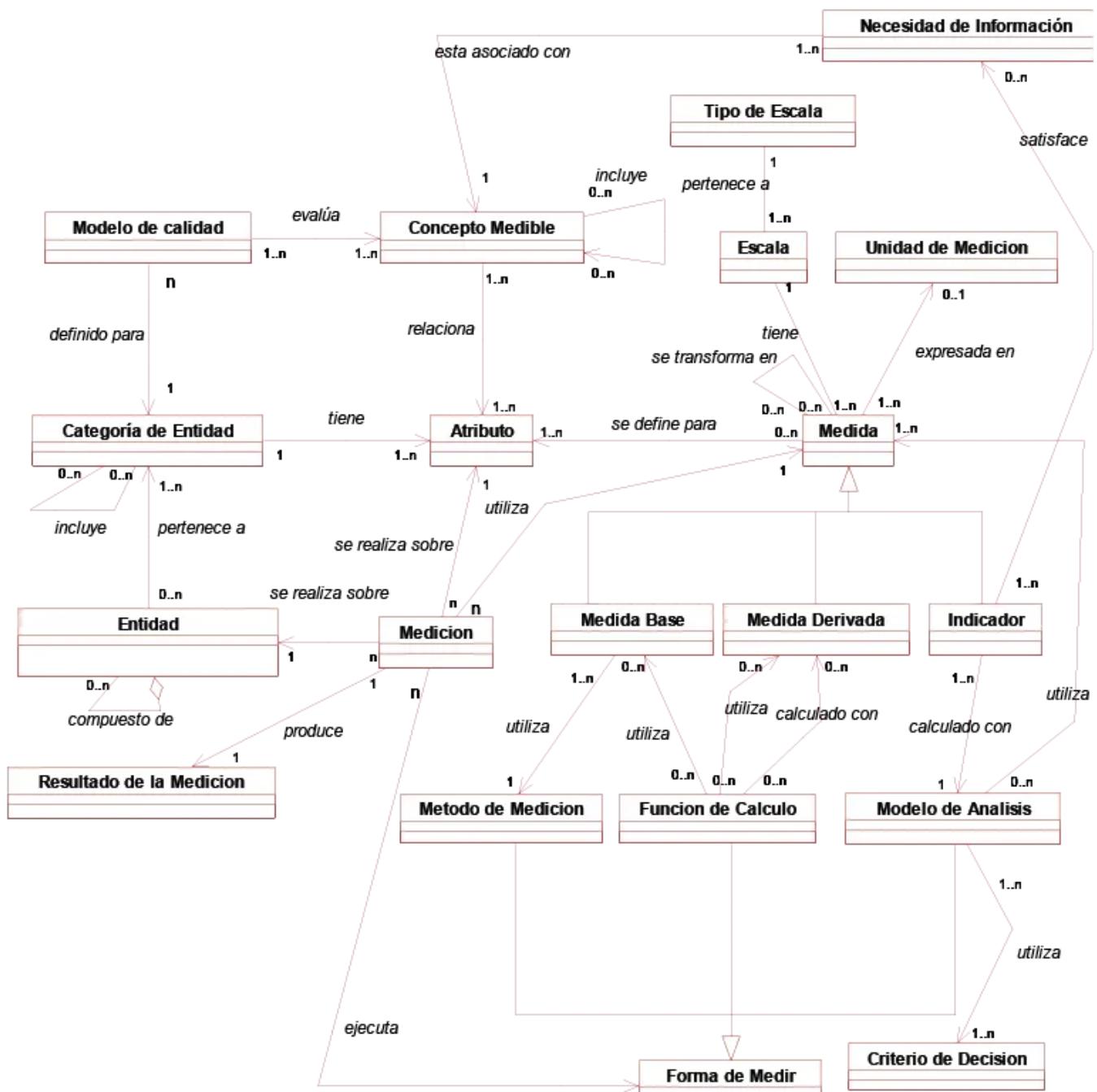


Figura 1.1 Conceptos de la Medición del Software y sus relaciones

Respecto a los atributos hay que hacer notar que suelen diferenciarse entre *atributos internos* y *externos*. Los atributos internos son aquellos que pueden ser medidos de una entidad sin necesidad de evaluar el comportamiento externo de dicha entidad. Ejemplos de atributos internos son: tamaño y complejidad de código fuente, que pueden ser evaluados sin necesidad de ejecutar el código. Los atributos externos son mediciones sobre cómo una entidad está relacionada con el entorno, como por ejemplo los atributos calidad y estabilidad de requisitos, en cuyo caso es necesario ejecutar el código para obtenerlos. Estos atributos son mucho más difíciles de evaluar que los atributos internos y la necesidad de disponer de mediciones de atributos internos para obtener el valor de atributos externos es bastante clara. Por ejemplo, para evaluar el atributo externo calidad del software es necesario conocer atributos internos como por ejemplo el número de fallos obtenidos en la actividad de pruebas.

Por otro lado, hay que destacar que la medición se puede apoyar en **modelos de calidad**, entendidos como “*Un conjunto de conceptos medibles y relaciones entre ellos que proporciona la base para especificar requisitos de calidad y evaluar la calidad de las entidades de una determinada categoría de entidad*”, como los modelos propuestos por la norma ISO/IEC 25010.

La **medida** o **métrica** es “la forma de medir (método de medición, función de cálculo o modelo de análisis) y la escala de medición”. Así, la medida o métrica “líneas de código” puede ser definida para realizar mediciones del “tamaño” de un “módulo en C” y para realizar mediciones del “tamaño” de un “programa en Java”. La **escala** es “un conjunto de valores con propiedades definidas”, como por ejemplo, el nivel de madurez de ISO/IEC 33000: 0, 1, 2, 3, 4, 5 (ordinal), el tamaño de un código software expresado en líneas de código: Conjunto de los números naturales (ratio), etc.

El **tipo de escala** indica la naturaleza de la relación entre los valores de la escala. En la teoría de la medición aplicada al software destacan cinco tipos principales de escalas:

- **Escala Nominal.** Es la escala más básica, que sitúa a las entidades en diferentes clases o categorías asignando al atributo un nombre. De este modo las clases se identifican únicamente mediante un número o símbolo que no puede ser interpretado salvo como un mero identificador. Por ejemplo, distinguir de forma nominal a los jugadores de un equipo de baloncesto por su dorsal. El jugador “30” no puede interpretarse como dos veces superior al jugador “15”.
- **Escala Ordinal.** Con esta escala los atributos pueden ser ordenados en rangos pero la distancia entre los mismos no es significativa. Por ejemplo, las respuestas a una encuesta podrían ser: “0 = totalmente en desacuerdo”, “1 = ni de acuerdo ni en desacuerdo”, “2 = de acuerdo”, “3 = muy de acuerdo”, “4 = totalmente de acuerdo”. En este caso, cuanto mayor es el valor hay un mayor acuerdo con lo planteado en la pregunta pero la distancia entre los valores 0 y 1 no tiene por qué ser igual que entre los valores 2 y 3.
- **Escala de Intervalo.** Este tipo de escala es como la ordinal pero con la diferencia de que la distancia entre los atributos sí tiene sentido. Por ejemplo, si se mide la temperatura en °C, la distancia entre 30 y 40 es la misma que entre 60 y 70. Sin embargo, en esta escala las comparaciones de tipo ratio no tienen sentido, como por ejemplo: 40° no es el doble de calor que 20° (aunque el valor sí es el doble). Si las mismas distancias entre valores tienen aproximadamente el mismo significado, entonces estamos hablando de una escala “quasi” intervalo.
- **Escala de Ratio,** que es la más útil en la medición del software, ya que preserva el orden, el tamaño de los intervalos y también los ratios entre las entidades. Además, tiene un punto fijo de referencia: el cero.

La escala debe comenzar en el 0 y se incrementa en pasos iguales.
 Además, con los valores de esta escala se pueden realizar las operaciones matemáticas de suma, resta, multiplicación y división. El peso es por ejemplo una variable de tipo ratio.

- **Escala Absoluta**, que es la más restrictiva de todas y es utilizada únicamente cuando sólo hay una forma posible de medir un atributo: la cuenta real. En general, los atributos evaluados mediante un método basado en contar el número de elementos son de este tipo de escala, como por ejemplo el número de defectos encontrados o el número de personas en un proyecto. Todas las operaciones aritméticas sobre el resultado en esta escala tienen sentido.

En la Tabla 1.1 se muestra un resumen de las propiedades de los tipos de escala (Maxwell 2002).

Tipo de Escala	¿Hay un orden significativo?	¿Las mismas distancias entre valores tienen el mismo significado?	¿El cálculo de ratios tiene sentido?
Nominal	No	No	No
Ordinal	Sí	No	No
Intervalo	Sí	Aproximadamente	No
Quasi-Intervalo	Sí	Sí	No
Ratio	Sí	Sí	Sí
Absoluta	Sí	Sí	Sí

Tabla 1.1. Resumen de propiedades de las escalas

La **unidad de medición** es “una cantidad particular, definida y adoptada por convención, con la que se pueden comparar otras cantidades de la misma clase para expresar sus magnitudes respecto a esa cantidad”

particular”. Ejemplos de unidades de medición serían: Líneas de código, Páginas, Persona-mes, Número de módulos, Número de clases, etc.

Otro de los aspectos a destacar de la teoría de la medición es la distinción que establece entre **medidas base (o directas)** y **derivadas (o indirectas)**. Una **medida base** es “una medida de un atributo que no depende de ninguna otra medida, y cuya forma de medir es un método de medición”, por ejemplo, LCF (líneas de código fuente escritas), HPD (horas-programador diarias), CHP (coste por hora-programador, en unidades monetarias). Mientras que una **medida derivada** es una “medida que es derivada de otra medida base o derivada, utilizando una función de cálculo como forma de medir”. Ejemplos de medidas derivadas son: HPT (horas-programador totales = sumatorio de las HPD de cada día), LCFH (LOC por hora de programador), CTP (coste total actual del proyecto, en unidades monetarias, que es el producto del coste unitario de cada hora por el total de horas empleadas).

También se pueden distinguir los **indicadores** que son “medidas derivadas de otras medidas utilizando un modelo de análisis como forma de medir”, por ejemplo, PROD (productividad de los programadores).

En cuanto a las **formas de medir** se puede distinguir un **método de medición**, que es “la forma de medir una medida base, y que se puede definir como la secuencia lógica de operaciones, descritas de forma genérica, usadas para realizar mediciones de un atributo respecto de una escala específica”. Por ejemplo, contar líneas de código o anotar cada día las horas dedicadas por los programadores al proyecto. La forma de medir de las **medidas derivadas** es utilizando una **función de cálculo** (algoritmo o cálculo realizado para combinar dos o más medidas base y/o derivadas), como por ejemplo, $LCFH = LCF / HPT$, medida derivada definida en base a dos medidas base.

La forma de medir un indicador es mediante un **modelo de análisis**, que lo podríamos definir como “un algoritmo o cálculo realizado para combinar

una o más medidas (base, derivadas o indicadores) con criterios de decisión asociados”. Un **criterio de decisión** “estará compuesto por valores umbral, objetivos, o patrones, usados para determinar la necesidad de una acción o investigación posterior, o para describir el nivel de confianza de un resultado dado”, por ejemplo, LCFH / LCFHvm (valor medio de LOC por hora de programador) < 0'70 --> PROD = 'muy baja'.

La **medición** entonces será “el conjunto de operaciones que permite obtener el valor del **resultado de la medición** para un atributo de una entidad, usando una forma de medir”. Por ejemplo, la acción consistente en usar la forma de medir “contar el número de líneas de código” para obtener el resultado de la medición del atributo “tamaño” de la entidad “módulo nóminas.c”. Mientras que el resultado de la medición será la categoría o número asignado a un atributo de una entidad como resultado de una medición. Por ejemplo: 35.000 líneas de código, 200 páginas, 50 clases, 5 meses desde el comienzo al fin del proyecto, etc.

En definitiva, todo proceso de medición del software tiene como objetivo fundamental satisfacer necesidades de información. Un proceso de medición no puede obtener resultados útiles si estos no satisfacen alguna necesidad de información detectada en la empresa en la que se lleva a cabo. A partir de las necesidades de información se deben identificar las entidades y los atributos de dichas entidades que son candidatos a ser medidos.

Una vez identificados los atributos objeto de la medición se deben definir las métricas necesarias. En la definición general de una métrica se deben especificar aspectos como la unidad en la que se expresa, la escala a la que pertenece, el atributo o atributos para los que se define, etc.

La definición de las métricas se debe realizar a distintos niveles o con diferentes alcances, ya que resultaría excesivamente complejo definir de forma directa métricas a partir de las cuales se satisfagan las necesidades de información. Por ello, es fundamental definir en primer lugar métricas que se aplican directamente sobre las características de una entidad para evaluar

un determinado atributo. A partir de estas métricas directas se pueden definir métricas indirectas y finalmente se podrían definir métricas con el objetivo de proporcionar información útil para la toma de decisiones y, por lo tanto, más cercanas a satisfacer las necesidades de información.

Finalmente, se lleva a cabo el proceso de medición propiamente dicho, a partir de la definición de las métricas y de la caracterización de los atributos de las entidades objeto de la medición, mediante la realización de mediciones que como resultado obtienen resultados de medición.

Otros marcos conceptuales

ISO/IEC/IEEE 15939

En el anexo A de la norma ISO/IEC/IEEE 15939 (ISO/IEC, 2017b) se presenta el “*measurement information model*” que se subsume en su totalidad en la ontología de la medición presentada.

Este modelo distingue explícitamente dos tipos de métodos de medición: subjetivos, cuya cuantificación involucra al juicio humano, y objetivo, basado en reglas numéricas (que pueden ser implementadas por medios humanos o automáticos). Se incluyen además ejemplos de “constructos de medición” como el que se recoge en la Tabla 1.2, para el “progreso”.

Necesidad de información	Evaluar el estado de la actividad de testing
Concepto medible	Estado de la actividad
Entidades relevantes	<ol style="list-style-type: none">1. Plan/planificación2. Elementos del sistema terminados o en progreso
Atributos	<ol style="list-style-type: none">1. Elementos del sistema identificados en el plan2. Estado de los elementos del sistema
Medidas base	<ol style="list-style-type: none">1. Elementos del sistema planificados hasta la fecha2. Porcentaje finalizado del elemento del sistema

Necesidad de información

Evaluar el estado de la actividad de testing

Método de medición	<ol style="list-style-type: none">1. Contar el total de elementos del sistema que deberían estar terminados a la fecha2. Preguntar a los responsables el porcentaje de finalización de cada elemento del sistema
Tipo de método de medición	<ol style="list-style-type: none">1. Objetivo2. Subjetivo
Escala	<ol style="list-style-type: none">1. Enteros de cero a infinito2. Enteros de cero a cien
Tipo de escala	<ol style="list-style-type: none">1. Ratio2. Ordinal
Unidad de medida	<ol style="list-style-type: none">1. Unidad2. Porcentaje
Medida derivada	Progreso hasta la fecha
Función de medida	Sumar el estado de todos los elementos que deberían estar terminados según la planificación
Indicador	Estado expresado como ratio
Modelo	Dividir el progreso hasta la fecha por unidades planificadas
Criterio de decisión	El ratio resultante debería estar entre 0,9 y 1,1 para poder concluir que el proyecto sigue la planificación

Tabla 1.2. Ejemplo de “constructo de medición” de ISO/IEC/IEEE 15939

ISO/IEC 2502X

Las normas ISO/IEC 25020 y 25021 de la división de medición de la calidad de la familia ISO/IEC 25000 son las que, respectivamente, definen un modelo de referencia de la medición y los elementos de medida de la calidad.

El modelo de referencia de la medición de la calidad (QM-RM, Quality Measurement Reference Model) (ISO, 2019) describe la relación entre un modelo de calidad y la construcción de QM (medidas de calidad, *quality measures*) a partir de QME (elementos de medida de calidad, *quality measure elements*), mediante la aplicación de funciones de medición que son algoritmos utilizados para combinar las QME. Los elementos de medida de calidad (QME) pueden ser medidas base, medidas derivadas (a partir de varias medidas base).

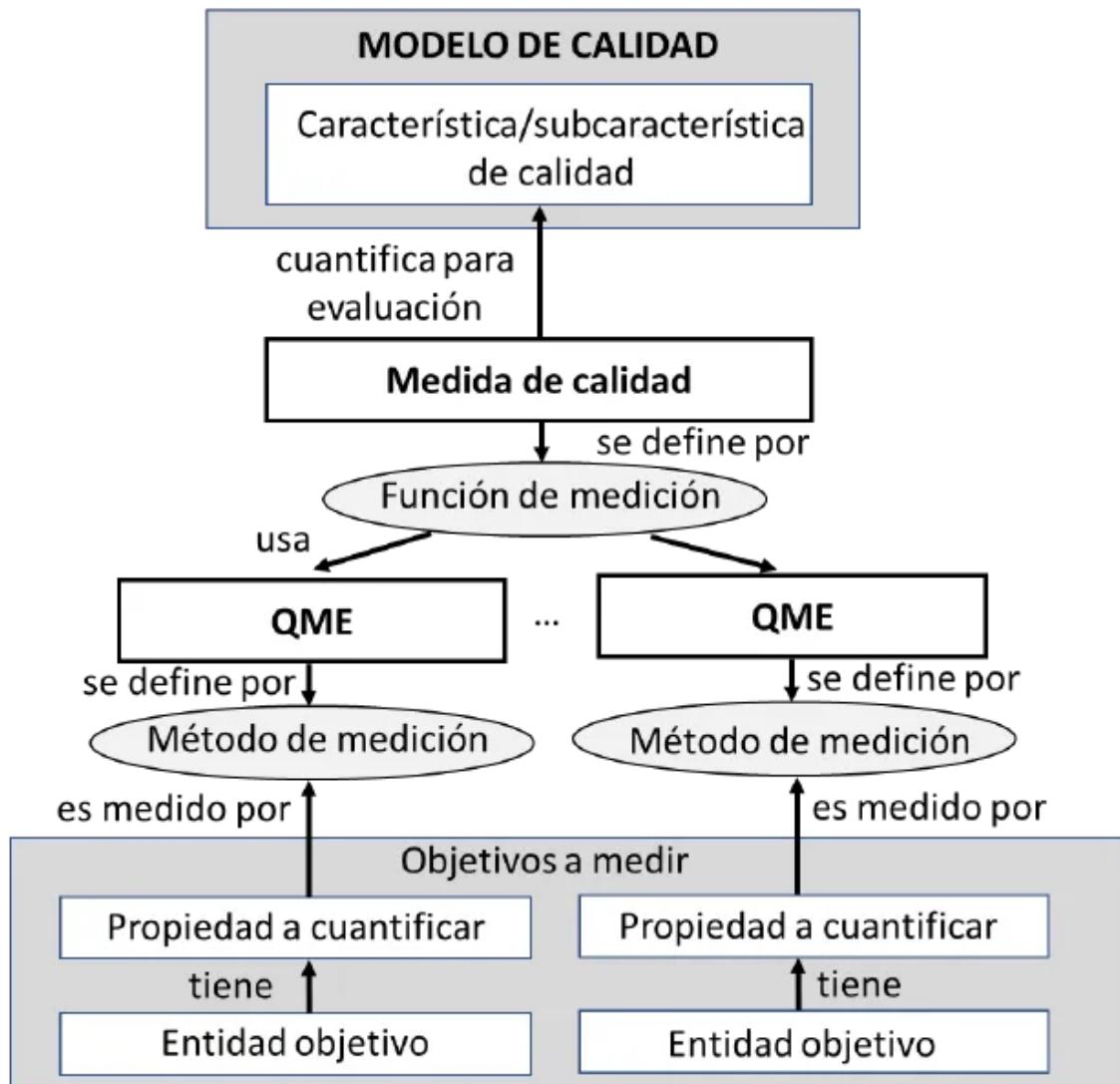


Figura 1.2 Modelo de referencia de medición de calidad (ISO, 2019)

La norma ISO/IEC 25021 (ISO, 2019) define un conjunto inicial de QME para ser utilizados en el marco de la familia ISO/IEC 25000, y proporciona directrices para que las organizaciones puedan desarrollar sus propios QME. Para describir los QME propone una tabla con las siguientes características:

- Nombre del QME
- Entidad objetivo
- Objetivos y propiedad a cuantificar
- Medidas de calidad relevantes
- Método de medición
- Lista de subpropiedades relacionadas con la propiedad a cuantificar
- Definición de cada subpropiedad
- Entrada para el QME
- Unidad de medición para el QME
- Reglas numéricas
- Tipo de escala
- Contexto del QME
- Procesos del ciclo de vida del software
- Restricciones a la medición

SMM DE OMG

El “*Structured Metrics Metamodel*” (SMM) del OMG (Object Management Group) (OMG, 2018), es un metamodelo para representar información de medición relacionada con cualquier modelo de información estructurada. Forma parte del ADM (*Architecture Driven Modernization*) pero soporta las necesidades de métricas de cualquier otro escenario tecnológico.

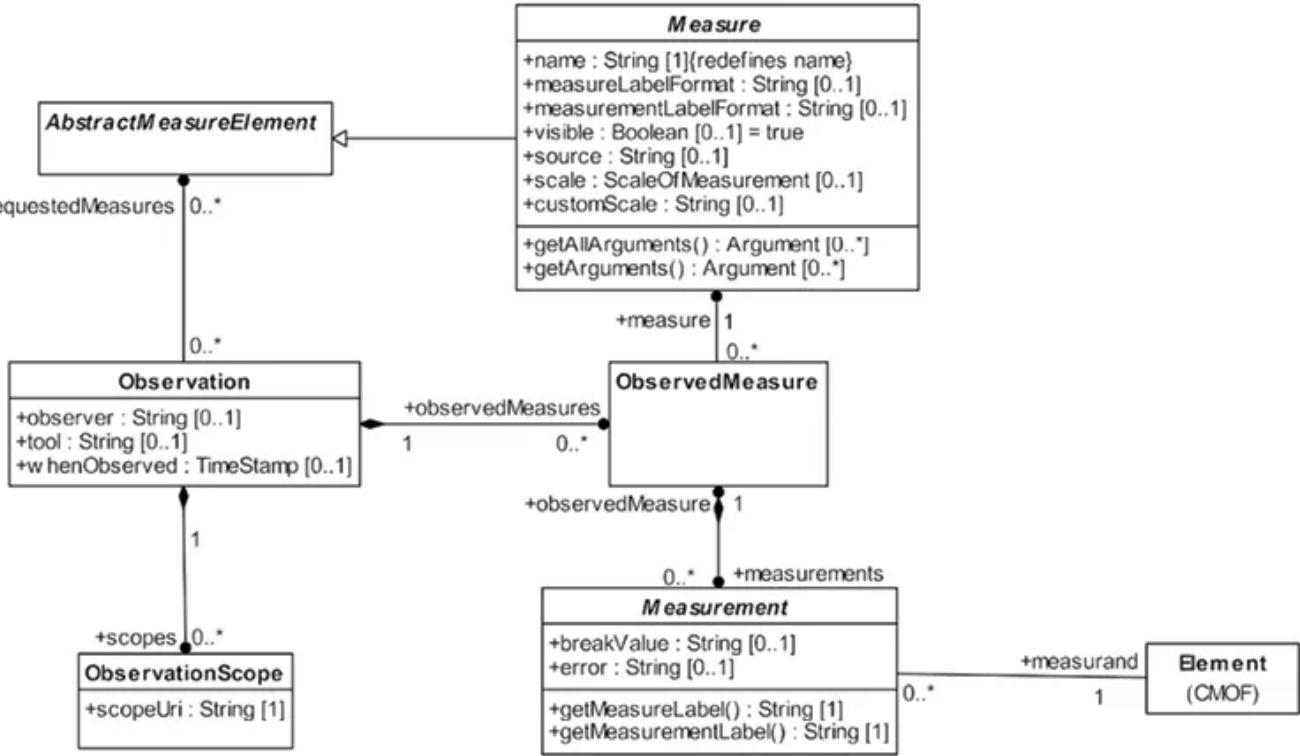


Figura 1.3 Constructores fundamentales del metamodelo SMM (OMG, 2018)

Como se puede ver en la Figura 1.3, en SMM la medición tiene un valor (*ObservedMeasure*) de acuerdo con una medida (entendida como proceso de medición), pudiendo ser el mensurando (elemento medible) cualquier elemento MOF, por lo que la medición se puede aplicar a cualquier elemento de cualquier metamodelo OMG.

La información contextual sobre la medida (quién, dónde, cuándo) se recoge en *Observation*, que permite diferentes usos de una medida en el mismo mensurando.

En el metamodelo se incluyen otras clases básicas como son: *SmmElement* (abstracta) que representa un elemento atómico constituyente de un modelo; *SmmModel Class* que representa el punto de entrada en el modelo y actúa de contenedor para todos los elementos del SMM; *SmmRelationship* (abstracta) que representa la asociación semántica entre elementos del SMM; *MeasureLibrary* que representa una biblioteca de medidas; *MeasureCategory*, que representa las categorías de medidas; *CategoryRelationship*, que representa la asociación entre la categoría de la

medida y los elementos de la medida; y el tipo *TimeStamp Primitive*, una combinación de fecha y tiempo.

Entre las categorías de medidas que se destacan en el SMM se encuentran: métricas de entorno (p.ej. número de programas, de líneas de código), de definición de datos (elementos de datos no utilizados, número de grupo de datos), de proceso de programas (McCabe, Halstead), de arquitectura (nivel medio de anidamiento de llamadas), funcionales (funciones definidas en el sistema), de calidad (fallos por día, tiempo medio de reparaciones), de desempeño (tiempo medio de respuesta en línea), o de aseguramiento de calidad.

Además, SMM incluye mecanismos para extensiones, medidas, medidas colectivas, otras medidas, mediciones, mediciones colectivas, mediciones nominadas y rescaladas, y observaciones.

LECTURAS RECOMENDADAS

- *CEM (2012). VIM. Vocabulario Internacional de Metroología, Conceptos fundamentales y generales y términos asociados. 3^a ed. Madrid, Centro Español de Metroología.*

Se trata de la traducción española del VIM, que establece el **vocabulario oficial de la metrología**.

- *Fenton, N. y Bieman, J. (2014). Software Metrics: A Rigorous and Practical Approach, 3^a Edición. Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series.*

Se trata de la última edición de un libro “clásico” que sentó las bases para la definición rigurosa de la medición en software.

- *Hubbard, D.W. (2014). How to Measure Anything. Finding the Value of “Intangibles” in Business 3^a ed. Wiley.*

En este libro se repasan temas clásicos de la medición, proponiendo un método que permite medir temas que aparentemente no se pueden

medir (p. ej., la flexibilidad organizacional).

2

<https://dogramcode.com/programacion>

GQM: GOAL QUESTION METRIC

VISIÓN GENERAL

El método GQM fue originariamente definido por Basili y Weiss (1984) y extendido posteriormente por Rombach (1990) como resultado de muchos años de experiencia práctica e investigación académica. El principio básico que subyace tras el método GQM es que la medición debe ser realizada, siempre, orientada a un objetivo. GQM propone definir un objetivo, refinarlo en preguntas y definir métricas que intenten dar información para responder a estas preguntas.

El método GQM se lleva a cabo en las siguientes fases (Solingen y Berghout, 1999):

- **Planificación**, durante la cual se selecciona, define, caracteriza y planifica un proyecto para la aplicación de la medición, obteniéndose como resultado un plan de proyecto.
- **Definición**, durante la cual se define y documenta el programa de la medición (objetivos, preguntas, métricas e hipótesis).

- **Recopilación de Datos**, en la que se reúnen los datos reales de la medición.
- **Interpretación**, en la que se procesan los datos recopilados respecto a las métricas definidas en forma de *resultados de medición*, que proporcionan respuestas a las preguntas definidas, a partir de las cuales se puede evaluar el logro del objetivo planteado.

Como puede verse en la Figura 2.1, la fase de planificación se lleva a cabo para satisfacer los requisitos básicos que permitan que un programa de medición GQM sea un éxito, para lo cual se incluyen aspectos de formación, implicación en la gestión y planificación de proyectos. Durante la fase de definición se elaboran productos que están principalmente basados en entrevistas estructuradas u otras técnicas de adquisición de conocimiento. En esta fase se identifica un objetivo, y todas las preguntas, métricas y expectativas (hipótesis) de las mediciones. Cuando se han completado todas las actividades de la medición, la medición real puede comenzar. Durante la fase de recopilación de datos se definen, llenan y almacenan en una base de datos una serie de formularios en los que se obtienen todos los datos de las mediciones. Finalmente, durante la fase de interpretación, se utilizan las mediciones para responder a las preguntas formuladas, y se utilizan las respuestas de nuevo para ver si se han logrado los objetivos planteados.

<https://dogramcode.com/programacion>

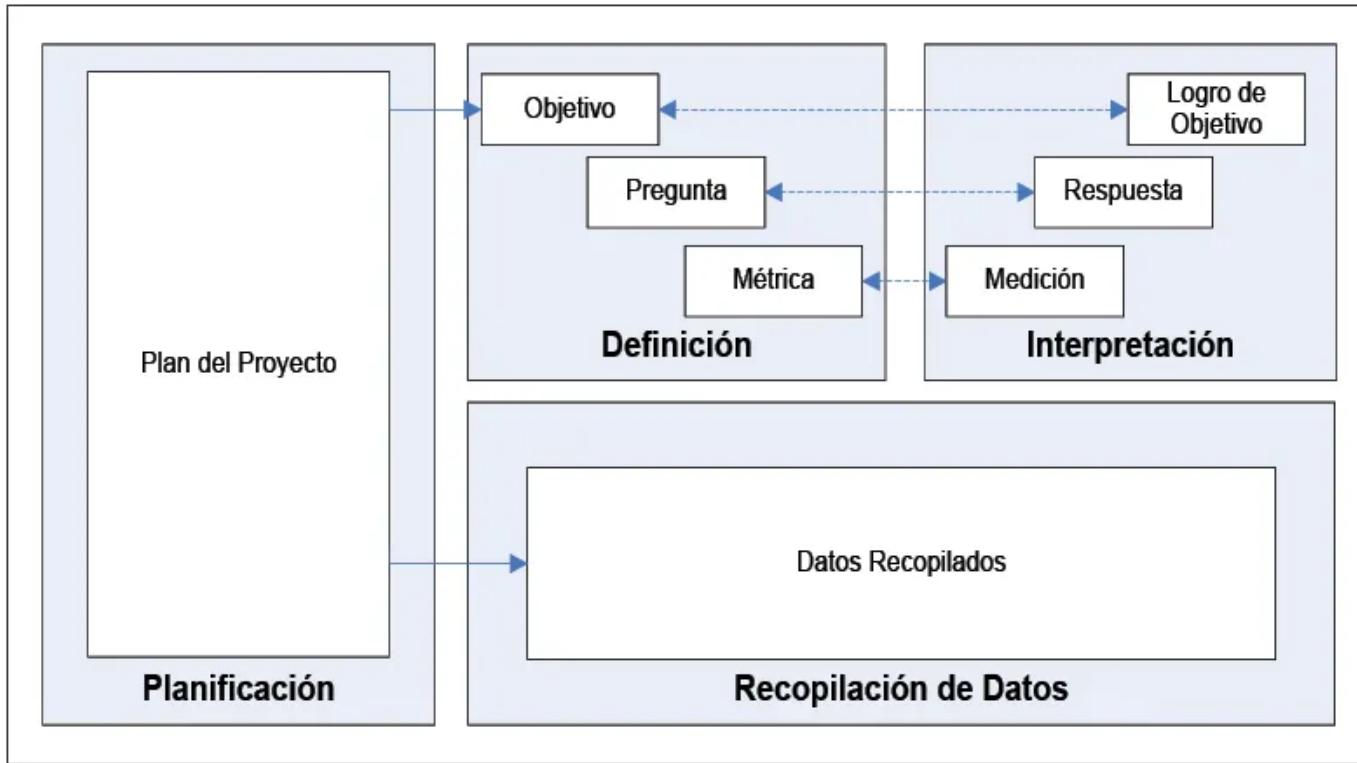


Figura 2.1 El Proceso GQM (Solingen y Berghout, 1999)

A continuación, se describen con mayor detalle las cuatro fases de GQM.

Fases de GQM

PLANIFICACIÓN

Los objetivos principales de esta fase son la recopilación de toda la información necesaria para un inicio exitoso del proyecto de medición, así como la motivación y preparación de los miembros de la organización para llevar a cabo el programa de medición. El plan del proyecto constituye el producto principal de esta fase, en el que se incluyen los documentos, procedimientos, calendarios y objetivos del programa de medición, así como un plan de formación de los desarrolladores implicados en el programa. El plan proporciona la base para el fomento y aceptación del programa de medición por parte de la dirección. Las etapas que componen la fase de planificación son:

- **Establecer el equipo GQM**, que es una etapa fundamental ante la necesidad de garantizar la continuidad de los programas de medición. En muchas ocasiones, cuando apremian los plazos de entrega de los productos, se dedica menos atención a las actividades del programa de medición, por lo que se requiere un equipo GQM que debería tener las siguientes cualidades: ser independiente de los equipos del proyecto y no ser “parte interesada” en los resultados de la medición; poseer suficiente conocimiento previo sobre los objetos de la medición; respetar a los miembros del proyecto cuando llevan a cabo las tareas del proyecto y reconocer que son ellos los que llevan a cabo las acciones de medición y mejora; tener una mentalidad de orientación a la mejora, incluso sobre sí mismos; y ser entusiastas para motivar a los miembros del proyecto. Los roles del equipo GQM son: gestor (*manager*), que es el responsable de dar continuidad en todo momento al programa de medición, “coach” que es el experto en GQM, e ingeniero de soporte (*support engineer*) a las actividades de medición. Las principales actividades del equipo de GQM son: planificar los programas de medición en el contexto de los proyectos de desarrollo; realizar las actividades de definición de la medición y desarrollar los entregables GQM; comprobar los datos recogidos por el equipo del proyecto y los datos disponibles del proceso; preparar la interpretación de los datos de la medición; e informar sobre el progreso del equipo de proyecto y de gestión y comunicar los resultados.
- **Seleccionar las áreas de mejora** de los productos o procesos, como podrían ser: problemas evidentes a los que se enfrenta la organización, áreas a mejorar identificadas tras una valoración de procesos, o áreas de mejora de productos en base a objetivos de negocio de alto nivel. Esta selección debe realizarse en función de los objetivos de negocio, y en especial en relación a los costes, tiempo, riesgos y calidad. Una vez seleccionada un área adecuada, el equipo

GQM debería considerar todos los detalles, incluyendo los problemas que podrían ocurrir; influencias externas como las personas implicadas, tecnologías, leyes, procesos y productos; y el conocimiento y experiencia previa en medición que tienen las personas que van a participar en el proyecto.

- **Seleccionar el proyecto de aplicación y establecer un equipo del proyecto.** El éxito de un programa de medición depende fundamentalmente de la voluntad, motivación y entusiasmo de los miembros del equipo de proyecto, ya que son ellos los que van a realizar las actividades de medición. Por ello, el equipo GQM debe hacer un esfuerzo para alinear los objetivos de la medición con las ideas de mejora del equipo del proyecto, para lo que deben controlar y estimular continuamente la dedicación del equipo del proyecto a las actividades de medición.
- **Crear el plan del proyecto,** actividad que se realiza una vez se ha establecido el equipo del proyecto y se han seleccionado las áreas de mejora. El plan del proyecto debería contener los siguientes elementos:
 - *Resumen Ejecutivo*, que presenta de forma resumida (en aproximadamente 20 líneas) el programa de medición.
 - *Introducción*, que presenta el alcance del programa de medición, así como la relación entre los objetivos de la mejora y los objetivos del proyecto de desarrollo de software.
 - *Calendario*, que incluye la planificación temporal, entregables, asignaciones de recursos y análisis coste-beneficio del programa de medición.
 - *Organización*, que describe las estructuras organizacionales, del proyecto y equipo GQM que son relevantes para el programa de medición.

- *Procesos de gestión*, que contiene prioridades, procedimientos de generación de informes de gestión así como actividades de control de riesgos.
- *Formación y promoción*, que presenta el plan para la formación de los miembros del equipo del proyecto y la comunicación de los resultados en la organización.
- **Formación y promoción**, ya que el equipo GQM debería organizar sesiones frecuentes de formación y promoción en las que se presenten de forma clara los objetivos de medición propuestos, los beneficios del programa de medición, el impacto del programa de medición en las actividades diarias del equipo de proyecto y las experiencias en otros proyectos u organizaciones. El objetivo es motivar y formar a los miembros del equipo del proyecto en la realización del programa de medición.

DEFINICIÓN

En esta fase se incluyen las actividades necesarias para definir formalmente el programa de medición y como resultado se obtienen los planes GQM, de medición y de análisis. Las etapas de la fase de definición son:

- **Definir los objetivos de la medición**, para lo que se consideran los objetivos de mejora del plan del proyecto definidos en la fase anterior. Como resultado se obtiene una definición formal y bien estructurada de los objetivos, para lo cual se utilizan plantillas como la que se muestra en la Tabla 2.1.

Analizar	el objeto de estudio bajo medición
Con el propósito de	entender, controlar o mejorar el objeto
Con respecto a	el enfoque de calidad del objeto en el que se centra la medición
Desde el punto de vista	la perspectiva de las personas que miden el objeto

de

En el contexto de

el **entorno** en el que la medición tiene lugar

Tabla 2.1. Plantilla de Definición de GQM

- **Revisar o elaborar los modelos de proceso software.** Estos modelos de procesos deben dar soporte a la definición de las mediciones. Si existen previamente en la organización, se deben revisar y si es necesario mejorarlos. Si no existen, el equipo GQM debe definir los modelos de procesos, que deben ser aprobados por el equipo del proyecto.
- **Realizar entrevistas GQM,** de forma que los miembros del equipo GQM puedan extraer de los miembros del equipo del proyecto toda la información relevante en relación a los objetivos de la medición. Para ello se realizan entrevistas individuales en las que para facilitar la comunicación se puede hacer uso de las hojas de abstracción (“*abstraction sheets*”), que incluyen aspectos básicos a considerar en las entrevistas GQM, tales como: *¿cuáles son las métricas para medir el objeto asociado a un determinado objetivo, de acuerdo a los miembros del proyecto?, ¿cuál es el conocimiento actual del miembro del proyecto respecto a estas métricas?, ¿qué factores externos pueden influenciar las métricas y de qué modo?*
- **Definir preguntas e hipótesis,** en base a los objetivos de la medición de forma que se dé soporte a la interpretación de los datos. De la misma forma que los objetivos se definen a un alto nivel de abstracción, las preguntas constituyen un refinamiento de los objetivos a un nivel más operacional. Con la respuesta a las preguntas planteadas, se debería poder concluir si se cumple un determinado objetivo. Para cada pregunta, las respuestas esperadas son formuladas como hipótesis que son comparadas en la fase de interpretación con los resultados reales de la medición.

- **Revisar preguntas e hipótesis**, para asegurar que se han formulado las preguntas e hipótesis correctas.
- **Definir las métricas**. Las métricas deben proporcionar la información cuantitativa que permita responder las preguntas planteadas de una forma satisfactoria. Por lo tanto, las métricas son el refinamiento de las preguntas en mediciones de los productos o procesos.
- **Comprobar consistencia y compleción de las métricas**, de forma que la definición de los objetivos, preguntas y métricas sea consistente y completa con respecto al objeto sujeto a medición.
- **Producir el plan GQM**, en el que se incluyen los objetivos, preguntas, métricas e hipótesis de un determinado programa de medición. Sirve como guía para la interpretación de los datos y para el desarrollo del plan de medición y análisis.
- **Producir el plan de medición**, en el que se incluye la definición formal, descripción textual y todos los resultados o valores posibles de las métricas directas así como el responsable de recoger dichos valores (programador, ingeniero, gestor del proyecto, etc.). También se incluye el momento en el que se debe tomar el valor de cada métrica directa y el medio (herramienta o formulario) que la persona encargada debe usar.
- **Producir el plan de análisis**, que es un documento en el que se simula la interpretación de los datos de acuerdo al plan GQM. Para ello se presentan simulaciones de los resultados de las métricas, así como gráficos y tablas en relación a los objetivos y preguntas planteadas. El plan de análisis pretende básicamente describir cómo la información relevante de la medición debe ser procesada con el fin de que pueda ser interpretada fácilmente por el equipo del proyecto

- **Revisar los planes**, que deben además ser aprobados por el equipo del proyecto antes de que comience la obtención de los datos reales de las mediciones.

En resumen, la definición de métricas con el método GQM se realiza mediante una aproximación arriba-abajo (Figura 2.2) en tres niveles: el nivel conceptual en el que se definen los objetivos (*goal*), el nivel operacional en el que se definen las preguntas (*question*) y el nivel cuantitativo en el que se definen las métricas (*metric*).

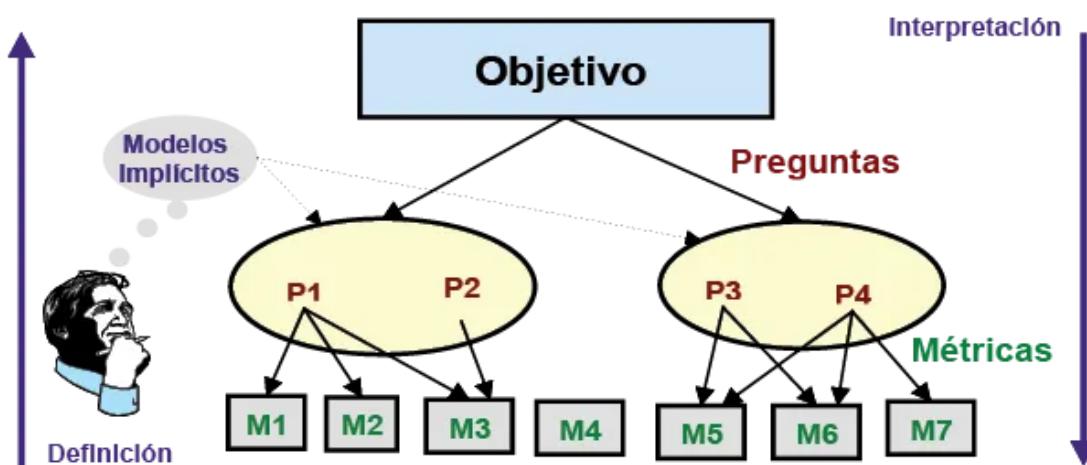


Figura 2.2 Fase de Definición de GQM (Basili y Weiss, 1984)

RECOPILACIÓN DE DATOS

Esta fase se inicia una vez se han completado todas las actividades de definición. Como resultado se obtiene una serie de formularios cumplimentados y almacenados en una base de datos.

Las principales etapas que componen esta fase son:

- **Formación e inicio de la obtención de datos**, que incluye:
 - **Periodo de Entrenamiento** (*Hold Trial*). Este periodo de prueba se lleva a cabo antes de comenzar la toma real de datos y durante el mismo se definen y prueban los procedimientos de recogida de datos así como las herramientas y formularios. En

esta actividad suelen intervenir dos personas como máximo (una al menos es preferible que sea un ingeniero senior) durante uno o dos días y el principal objetivo es evitar errores y detectar posibles mejoras a realizar en los procedimientos, herramientas o formularios.

- **Sesión de Inicio** (*Kick off*), durante la que todos los participantes en el programa de medición deben estar presentes. El principal objetivo es llegar a un acuerdo con el equipo del proyecto para el comienzo de la recogida de datos de la medición y se instruye a sus miembros en los procedimientos de recogida de datos, herramientas y formularios.
- **Recogida de Datos**, durante la que se llenan los formularios y se entregan de manera frecuente, preferiblemente de forma diaria, al equipo GQM. El equipo GQM comprueba la consistencia y corrección y almacena los formularios, estableciendo la base de métricas para el posterior establecimiento del sistema de soporte a la medición.

- **Construcción del sistema de soporte a la medición** (*Measurement Support System*, MSS). Este sistema constituye un elemento esencial del programa de medición siendo su base un conjunto de herramientas genéricas tales como hojas de cálculo, herramientas estadísticas, aplicaciones de bases de datos y herramientas de presentación. El MSS debe dar soporte a todas las actividades de medición, en las que se incluyen la obtención, almacenamiento, procesamiento, presentación y empaquetamiento de los datos de medición. El sistema MSS está formado por tres partes básicas:

- *Base de Métricas del MSS*, que contiene los datos recabados.
- *Hojas de Análisis del MSS*, que son los distintos tipos de presentación de los datos obtenidos respecto a diferentes niveles

de abstracción, que en orden ascendente son: datos sin procesar, datos procesados, y gráficos y diagramas. Cada hoja de análisis debe incluir: la descripción del objetivo y todas las preguntas derivadas del mismo (como en el plan GQM); así como todos los datos necesarios para responder las preguntas planteadas de una forma satisfactoria con respecto a los objetivos e hipótesis.

- *Transparencias de Análisis del MSS*, que son las transparencias de presentación que se mantienen de forma que cualquier cambio de las hojas de análisis produzca su actualización inmediata.

INTERPRETACIÓN

La fase de interpretación utiliza los datos tomados en la medición para responder las preguntas planteadas y, de esta forma, para identificar si se alcanzan o no los objetivos. Las etapas incluidas en esta fase son:

- **Preparación de las sesiones de realimentación**, en la que los miembros del equipo GQM deberían preparar todo el material necesario, como: hojas de análisis, diapositivas de presentación, material adicional, etc.
- **Sesiones de realimentación**. Durante estas reuniones se deben debatir los resultados de la medición y se suelen celebrar cada seis u ocho semanas con una duración típica de una hora y media a dos horas. Durante estas reuniones el equipo del proyecto, como expertos en el objeto bajo medición, debe analizar los resultados y obtener conclusiones y acciones a realizar. Para ello se centran en: evaluar puntos de acción de sesiones previas; analizar e interpretar los datos recogidos en la medición respecto a los objetivos y preguntas del plan GQM y obtener conclusiones y traducir estas conclusiones en acciones concretas. Para ello los miembros del equipo del proyecto deben adoptar un enfoque constructivo y dirigido por objetivos.

- **Generación de informes de interpretación de los resultados de la medición.** Al final de cada sesión de realimentación el equipo GQM elabora un informe en el que se incluyen todas las observaciones, interpretaciones, conclusiones y puntos de acción relevantes formulados. Este informe se distribuye a los miembros del equipo del proyecto.
- **Análisis de costes y beneficios del programa de medición.** El factor fundamental del éxito de un programa de medición es el logro de los objetivos planteados. Sin embargo, es también necesario incluir en el informe final un análisis de costes/beneficios.

En Solingen y Berghout (2001) se profundiza algo más tanto en las sesiones de realimentación, identificando los factores más importantes para facilitar el aprendizaje, a saber:

- Clima de “cultura de aprendizaje”, con flujo libre de información, comunicación abierta, compartición de problemas y lecciones aprendidas, etc.
- Búsqueda de conocimiento, que debe ser continua y concernir a cualquier conocimiento que pueda ser relevante o aplicable en situaciones específicas.
- Información sobre el contexto y el estado actual del sistema.
- Aprendizaje de equipo.
- Modelado del sistema bajo control (sobre todo de las relaciones entre los requisitos de los productos y procesos).
- Posibilidades de control.
- Dirección implicada con el aprendizaje y la motivación del personal.
- Definición explícita de objetivos.

- Monitorización de la “brecha” (gap) de desempeño, entre la situación actual y la situación objetivo.

Estos autores también proponen un modelo de esfuerzos a aplicar en GQM (véase Tabla 2.2), señalando la siguiente estructura de costes para una aplicación GQM rutinaria:

- Casi el 30% del esfuerzo se emplea en la definición del programa de medición, mientras que el 70% restante se dedica a su continuación, sobre todo en las sesiones de realimentación.
- El 70% del esfuerzo lo lleva a cabo el equipo GQM, mientras que sólo el 30% lo dedica el equipo del proyecto.
- El esfuerzo dedicado por el equipo del proyecto al programa de medición es menor que el 1% de su esfuerzo total de trabajo.
- Un total de 3 meses-persona, distribuido en un año, es el esfuerzo típico en un programa GQM.

Tarea	Equipo GQM	Director	1 ingeniero	10 ingenieros	Total
Planificación del programa GQM	4	2	-	-	6
Identificar y definir objetivos GQM	8	1	1	10	19
Realizar entrevistas GQM	40	2	2	8	50
Desarrollar plan GQM	40	1	1	6	47
Recoger datos	24	1	1,5	15	40
Análisis e interpretación de datos (por sesión de realimentación)	48	2	2	20	70

Análisis e interpretación de datos (5 sesiones de realimentación)	240	10	10	100	350
Total	356	17	15,5	139	512

Tabla 2.2. Ejemplo de modelo de esfuerzo para la aplicación rutinaria de GQM (horas/persona) (Solingen y Berghout, 2001)

Sin embargo, la primera vez que se usa GQM el reparto de los esfuerzos suele variar de la siguiente manera:

- El programa inicial necesita 8 meses-personas de esfuerzo.
- Aproximadamente el 50% del esfuerzo total se emplea en la definición del programa de medición.
- El esfuerzo invertido por el equipo de desarrollo suele ser el doble (menor del 2% del total) que en las aplicaciones sucesivas.

Extensiones a GQM

Existen en la literatura muchas propuestas de extensiones al modelo GQM, en este apartado resumimos algunas de ellas.

GOAL QUESTION INDICATOR METRIC (GQ(I)M) Y GOAL-DRIVEN SOFTWARE MEASUREMENT (GDSM)

La metodología GQ(I)M identifica y define métricas software que dan soporte al negocio de la empresa, la mejora de sus procesos y los objetivos de sus proyectos, asegurando la relevancia y trazabilidad de los objetivos respecto a los datos obtenidos. GQ(I)M comparte muchas similitudes con la metodología anteriormente descrita GQM, salvo en el aspecto de que añade soporte explícito a los indicadores. Por ello, el artefacto más relevante de esta metodología es la “Plantilla de Indicadores”, que es utilizada para definir de forma precisa el “quién”, “qué”, “dónde”, “cuándo”, “por qué” y

“cómo” de un indicador, así como para documentar el alineamiento del mismo con los objetivos de la organización. Todo ello garantiza disponer de una colección consistente de métricas a la hora de construir un indicador y proporciona elementos adicionales para asegurar una interpretación consistente del propio indicador (Goethert y Siviy, 2004). El proceso que sigue la metodología GQ(I)M es el propuesto por el SEI en su enfoque “Goal-Driven Software Measurement” (Park et al., 1996). La metodología está formada por diez pasos organizados en torno a tres conjuntos generales de actividades (Park et al., 1996; Zubrow, 1998):

Identificación de Objetivos

Que se divide en los siguientes pasos (véase Figura 2.3):

1. **Identificar los objetivos de negocio.** En este primer paso se deben identificar y priorizar los objetivos que dirigen los esfuerzos de la organización. Como resultado se obtiene una lista de objetivos ordenada según su prioridad.
2. **Identificar lo que se quiere conocer o aprender.** Una vez identificados los objetivos de negocio, el siguiente paso es comenzar a identificar lo que le gustaría saber a la organización con el fin de entender, valorar, predecir o mejorar las actividades relacionadas con la consecución de los objetivos. Para ello se formulan preguntas tales como “*¿Qué actividades tengo que gestionar o realizar?*” y se completan sentencias del tipo “*Para hacer esto, necesitaré...*”. Ello implica en este paso la traducción de los objetivos de negocio en declaraciones a nivel operacional. Los objetivos son enlazados con el conocimiento obtenido de los procesos de negocio y estrategias de la organización. Las preguntas relacionadas con los objetivos planteados son estructuradas en forma de entidades (productos de trabajo o actividades) y atributos (tamaño, esfuerzo o calidad) asociados con los procesos de la organización. En muchas ocasiones la descripción de los procesos

de trabajo de la organización no es explícita y se encuentra más bien como un modelo mental en la misma. En todo caso es de gran relevancia identificar los productos de trabajo, actividades y otras entidades que puedan ofrecer oportunidades de mejora.

3. Identificar los subobjetivos. Los subobjetivos proporcionan un refinamiento a nivel operacional de los objetivos generales. Se obtienen analizando las similitudes y aspectos comunes de las preguntas planteadas sobre las entidades, agrupando las mismas en función de los aspectos que tratan de resolver. Ello permite identificar más claramente la relación entre los resultados obtenidos (respuestas a las preguntas) y los objetivos.

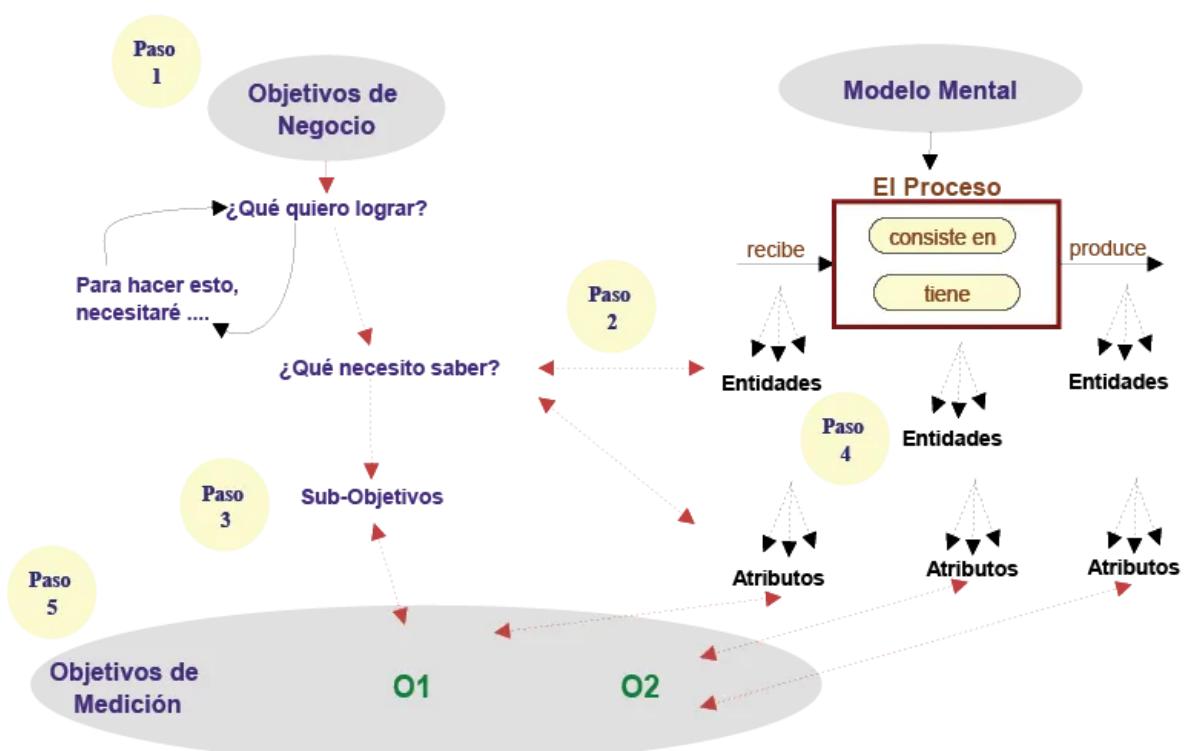


Figura 2.3 Identificación de Objetivos en GQ(I)M (Park et al., 1996)

4. Identificar las entidades y atributos relacionados con los sub-objetivos. En este paso se utilizan las preguntas para refinar el modelo mental del proceso y sus entidades y atributos asociados.

Esto permite establecer un conjunto bien definido de objetivos de negocio que son utilizados para el arranque del proceso GQ(I)M propiamente dicho. Este paso comienza seleccionando las preguntas que se consideran relevantes y que suelen estar asociadas con los subobjetivos de mayor prioridad. Una vez que se tiene una lista de preguntas, es necesario identificar las entidades implicadas y sus atributos. La cuantificación de estos atributos debe permitir obtener respuestas a las preguntas planteadas. Este paso es iterativo, lo que puede conducir al refinamiento de las cuestiones y subobjetivos a medida que se mejora el modelo mental del proceso.

5. Formalizar los objetivos de negocio. En este paso se traducen los objetivos de negocio en objetivos de medición, que enlaza el propósito y las perspectivas de los objetivos del negocio con las posibilidades de medición que existen en la organización de acuerdo a sus procesos de trabajo. Además, el objetivo de medición expresa los factores del entorno que es importante entender por todos aquellos encargados del diseño y realización de las actividades de medición y análisis. Para estar bien estructurados los objetivos de medición deben incluir: el objeto de interés (entidad); el propósito, la perspectiva y una descripción del entorno y restricciones. El propósito de la medición puede ser: entender, predecir, planificar, controlar, comparar, valorar o mejorar algún aspecto de calidad o productividad del objeto o entidad. La perspectiva identifica quién o quiénes son los interesados en los resultados de la medición. La información del contexto, o entorno, ayuda en la interpretación de los resultados de la medición.

Definición de Indicadores

Que se pueden descomponer en:

1. Identificar preguntas cuantificables y los indicadores relacionados.

relacionados. En este paso se identifican las preguntas e indicadores a partir de cada uno de los objetivos de medición planteados. Los indicadores representan los productos obtenidos en las actividades de medición y son utilizados por los directores de proyectos y profesionales como fuente de información de soporte para la toma de decisiones. A la hora de diseñar un indicador hay que considerar aspectos tales como la frecuencia de toma de datos, el tiempo requerido para generar el indicador, la necesidad de datos históricos, etc. En este sentido es muy útil el uso de plantillas que faciliten la definición de indicadores. Otro aspecto importante para facilitar que las organizaciones entiendan claramente si han alcanzado sus objetivos es distinguir los tipos de indicadores que se pueden definir. Goethert y Siviy (2004) distinguen tres tipos de indicadores:

- *Indicadores de éxito.* Estos indicadores se construyen a partir de criterios de éxito y se usan para determinar si se han alcanzado los objetivos (por ejemplo, de calidad o cuota de mercado). Los cuadros de mando integrales (*balanced scorecards*) suelen contener este tipo de indicadores.
- *Indicadores de progreso.* Estos indicadores se utilizan para realizar el seguimiento del progreso en la ejecución de las tareas definidas. Por ejemplo, las técnicas de “valor añadido” (*earned value*) o los diagramas de Gantt permiten construir buenos indicadores de este tipo. El cumplimiento de los valores de este tipo de indicador significará que la ejecución de las tareas se está llevando a cabo con éxito pero no garantiza la consecución de los objetivos de negocio, aunque un fallo en este indicador puede significar un problema importante para conseguir dichos objetivos.

- *Indicadores de análisis.* Este tipo de indicadores es utilizado para ayudar en el análisis de las salidas producidas por las tareas. Un indicador que representa el número y tipo de defectos detectado en cada fase del desarrollo es un ejemplo de este tipo de indicadores.

2. Identificar los elementos de datos. Los indicadores reflejan los elementos de datos que son necesarios. En esta etapa se identifican estos elementos, lo que significa que no es necesario aún que se definan las métricas, tarea que se realiza en el siguiente paso.

3. Definir las métricas. Una vez identificados los elementos de datos, hay que definir las métricas necesarias que permitan obtener respuesta a las preguntas planteadas. La definición de las métricas es un paso clave para obtener una interpretación adecuada de los datos recogidos y debe realizarse teniendo en mente el propósito del indicador o indicadores. Para facilitar una definición no ambigua de las métricas, el SEI ha propuesto una serie de marcos de trabajo de la medición con listas de comprobación para métricas que evalúan atributos como el tamaño, esfuerzo, consecución de hitos y defectos (Park, 1992; Florac, 1992; Goethert, 1992).

En la Figura 2.4 se muestra el esquema general para la definición de indicadores.

Crear un plan de acción

Este plan de acción se compone de los siguientes pasos:

1. Identificar las acciones a implementar. Hasta ahora se dispone de la definición de los indicadores y métricas que pueden dar respuesta a las preguntas planteadas en función de los objetivos de negocio. En este paso es el momento de analizar la

situación actual en la organización con respecto a las necesidades de información planteadas. Es necesario identificar las fuentes de información existentes en la organización ya que los elementos de datos requeridos podrían encontrarse en una gran diversidad de fuentes que incluyen planes de proyectos, sistemas de seguimiento de defectos, de gestión de configuración, de generación de informes de esfuerzo, etc. Del mismo modo, hay que hacer un análisis de los datos que son necesarios y no están disponibles en la organización en el que se valore la cantidad de esfuerzo que requiere su obtención. En este aspecto se considera si son necesarias nuevas herramientas, formularios o incluso formación para obtener los datos. En este paso también se deben priorizar los datos respecto a los indicadores de los que dependen. Por ello para cada elemento de datos se debe determinar su estado respecto de si existe una explícita definición de una métrica para dicho elemento de datos, si se han determinado los puntos en el proceso en el que se realizarán las mediciones y su frecuencia, si hay formularios y procedimientos para recoger y registrar los datos, quién tomará dichos datos, cómo se analizarán, si hay herramientas de soporte, etc.

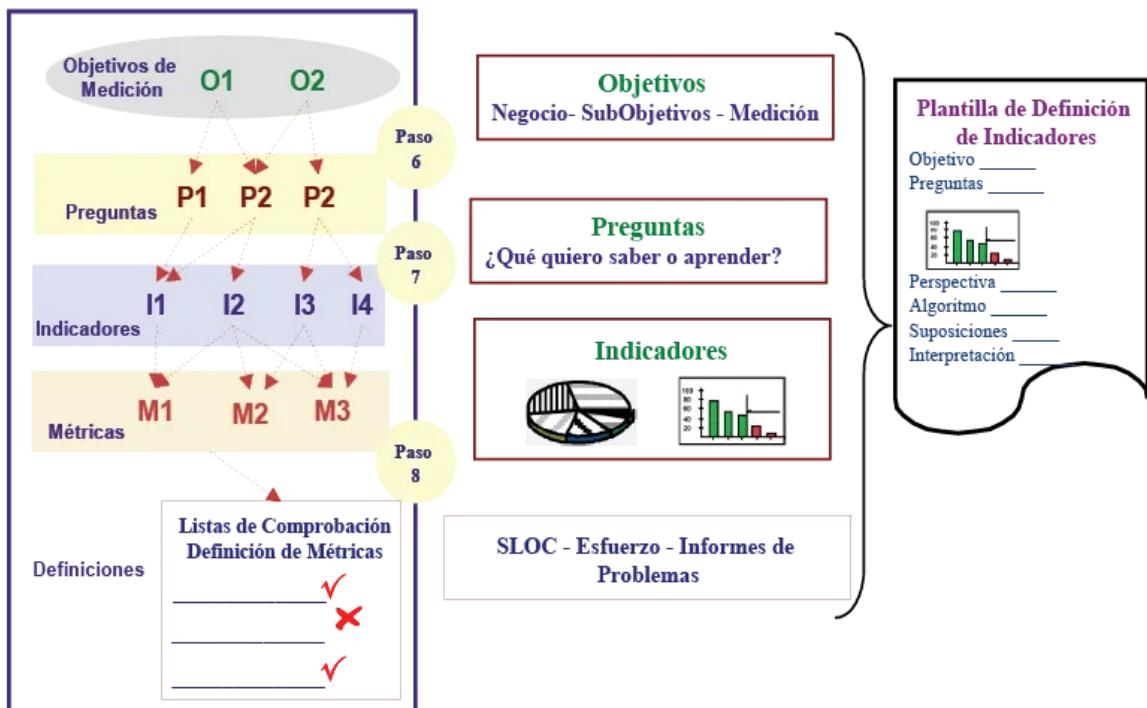


Figura 2.4 Definición de Indicadores en GQ(I)M (adaptado de Goethert y Siviy (2004) y Park et al. (1996))

2. Preparar un plan de acción. Una vez que se ha realizado el análisis necesario y se conocen los datos requeridos y las actividades de medición a llevar a cabo para obtenerlos, es el momento de definir el plan en el que se incluyan las acciones concretas a llevar a cabo para satisfacer las necesidades de información planteadas.

Plantilla para la definición de indicadores

La plantilla para la definición de indicadores constituye el artefacto clave de la metodología GQ(I)M. Las organizaciones frecuentemente no obtienen los beneficios potenciales de un buen programa de medición debido a las inconsistencias en la construcción e interpretación de los indicadores derivados de los datos de la medición.

Para ello es importante disponer de una plantilla para su definición y de una guía que facilite la utilización de la misma (Goethert y Siviy, 2004). También ayuda para asegurar la recopilación consistente de las métricas necesarias para obtener el indicador, así como los criterios necesarios para la interpretación de las métricas obtenidas. Para la definición de un indicador la plantilla incluye los siguientes campos:

- Objetivo del indicador: el objetivo o propósito del indicador.
- Preguntas: la lista de preguntas que el usuario del indicador intenta responder con su definición.
- Representación gráfica del indicador.
- Perspectiva o punto de vista, es decir, la descripción de la audiencia para la que se ha definido el indicador.
- Entradas: la lista y definiciones de las métricas requeridas para construir el indicador.
- Algoritmos: la descripción de los algoritmos usados para construir el indicador a partir de las métricas definidas.
- Suposiciones: la lista de suposiciones sobre la organización, sus procesos, modelo de ciclo de vida y todos aquellos datos que sean importantes para obtener y usar el indicador.
- Información de toma de datos: en la que se indica cómo, cuándo, con qué frecuencia, quiénes, etc. reúnen los elementos de datos requeridos en la construcción del indicador.
- Información de generación de informes de datos: información sobre quién es responsable de generar los informes de los datos, para quiénes y la frecuencia de almacenamiento, recuperación y seguridad de los datos.
- Análisis e Interpretación de los resultados: información sobre cómo analizar e interpretar el indicador.

Las organizaciones pueden adaptar la estructura de esta plantilla a sus entornos particulares ya que la plantilla propuesta por el SEI incluye los campos que se consideran comunes para la definición de indicadores.

Algunos ejemplos de aplicación de la plantilla de indicadores pueden consultarse en Goethert y Siviy (2004).

V-GQM

Olsson y Runeson (2001) proponen V-GQM (*Validating Goal/Question/Metric*) que añade una perspectiva de ciclo de vida a GQM, creando un proceso que comprende varios estudios GQM. Así, proponen dividir las métricas en cuatro categorías:

- No disponibles.
- Extendidas (si se recogieron más métricas que las sugeridas en el plan GQM).
- Generalizables (si se pueden utilizar para responder más preguntas que las planteadas en el plan). Por ejemplo, la métrica de proceso “distribución de defectos a lo largo del tiempo” (con la que se pretende identificar dónde intensificar los esfuerzos de verificación y validación) podría servir para informar sobre la densidad de defectos del producto.
- Suficientes.

Después de esta validación de las métricas se lleva a cabo un análisis de las preguntas, para ver cuáles se pueden mejorar y qué nuevas preguntas se derivan. Posteriormente, se refinan los objetivos y se hacen evolucionar.

GQM/MEDEA

Briand et al. (2002) proponen un proceso de medición denominado GQM/MEDEA (*GQM/Metric Definition Approach*), utilizable como una guía práctica para diseñar y reutilizar métricas técnicamente sólidas y útiles.

El objetivo de este proceso es la construcción de sistemas de predicción, es decir, modelos que establecen una correspondencia entre los atributos software.

Los cuatro pasos de alto nivel de este proceso son:

- Determinar el estudio empírico, refinando los objetivos de la organización en objetivo de medición, basándose en el conocimiento de entorno que proporcionan los equipos del proyecto y la factoría de experiencias.
- Definir las medidas para los atributos independientes.
- Definir las medidas para los atributos dependientes.
- Refinar y verificar las hipótesis.

En esta propuesta, los autores proporcionan además un modelo conceptual que puede ser utilizado para implementar el esquema de un repositorio que contenga todo el conocimiento relevante sobre medición.

MULTIVIEW FRAMEWORK

Baldassarre et al. (2003) proponen un método, denominado “marco multivista” (*multiview framework*), para diseñar un plan de medición de acuerdo con el modelo GQM y estructurarlo de manera que mejore su usabilidad. Pretenden que los planes de medición sean comprensibles, esto es, que permita que las medidas se interpreten correctamente dependiendo de los valores de las métricas e independientemente de quien las interpreta; y que sean eficientes, es decir, que el esfuerzo para interpretar las medidas sea relativamente pequeño a pesar de su exactitud.

Este marco presenta cuatro pasos:

- Definición del proyecto. En la que se identifican los procesos, entregables, actividades de gestión de proyectos y artefactos

correspondientes; y actividades y artefactos para evaluar la adecuación de los objetivos de inversión.

- Determinación de objetivos, procesos, productos, gestión de proyectos y adecuación de inversión.
- Validación cruzada, entre los objetivos y los procesos, productos, gestión de proyectos y adecuación de inversión. Con el fin de medir la comprensibilidad del plan de medición se utilizan dos métricas: el número de dependencias y la densidad de dependencias.
- Verificación de la complejidad de la interpretación, para lo que se define una tabla de decisión para cada objetivo del plan de medición.

GQM EFICIENTE

Berander y Jönsson (2006) proponen una aproximación estructurada para identificar mediciones usables y útiles para diferentes procesos, tal y como se muestra en la Figura 2.5.

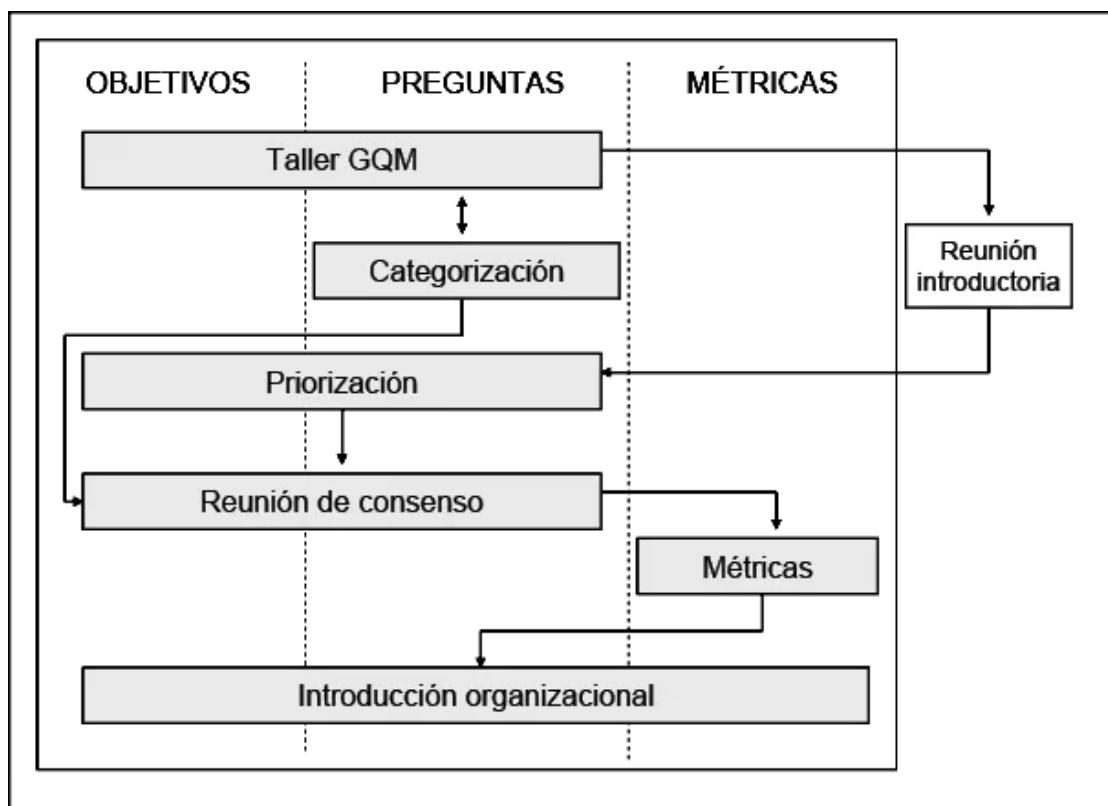


Figura 2.5 Proceso GQM eficiente (Berander y Jönsson, 2006)

Como se puede ver en la Figura 2.5, este método consta de los siguientes pasos:

- Taller (*workshop*) GQM, con el fin de identificar objetivos y cuestiones interesante para la organización.
- Categorización, de las cuestiones respecto a sus características, con el fin de incluir cuestiones relativas a varias dimensiones.
- Reunión de introducción a la tarea de formular cuestiones, sobre cómo funciona GQM, etc.
- Priorización, de objetivos y cuestiones.
- Reunión de consenso, con el fin de determinar qué objetivos y cuestiones se incluyen, discutir los resultados y compartir experiencias (transferir conocimiento).
- Métricas, identificadas a partir de los objetivos y cuestiones escogidos.
- Introducción organizacional, que trata sobre la documentación y comunicación de los resultados a las partes interesadas (*stakeholders*).

GQM DFMS

Gencel et al. (2013) proponen un marco de trabajo de soporte a la selección de métricas basado en GQM (véase Figura 2.6). El objetivo es facilitar a las organizaciones la creación de programas de medición sostenibles en el tiempo, teniendo en cuenta la gran cantidad de objetivos y métricas que pueden surgir en programas de medición complejos y las limitaciones de recursos para la medición que pueden tener ciertas organizaciones.

El principio fundamental de GQM-DSFMS es optimizar de forma continua la selección de métricas dado un determinado presupuesto del programa de medición mediante la priorización de los objetivos y el seguimiento de la trazabilidad de las métricas y objetivos. La prioridad de las metas redunda en la importancia de las métricas, que tienen un coste asociado. Estas prioridades y costes se usan para seleccionar de forma óptima las métricas.

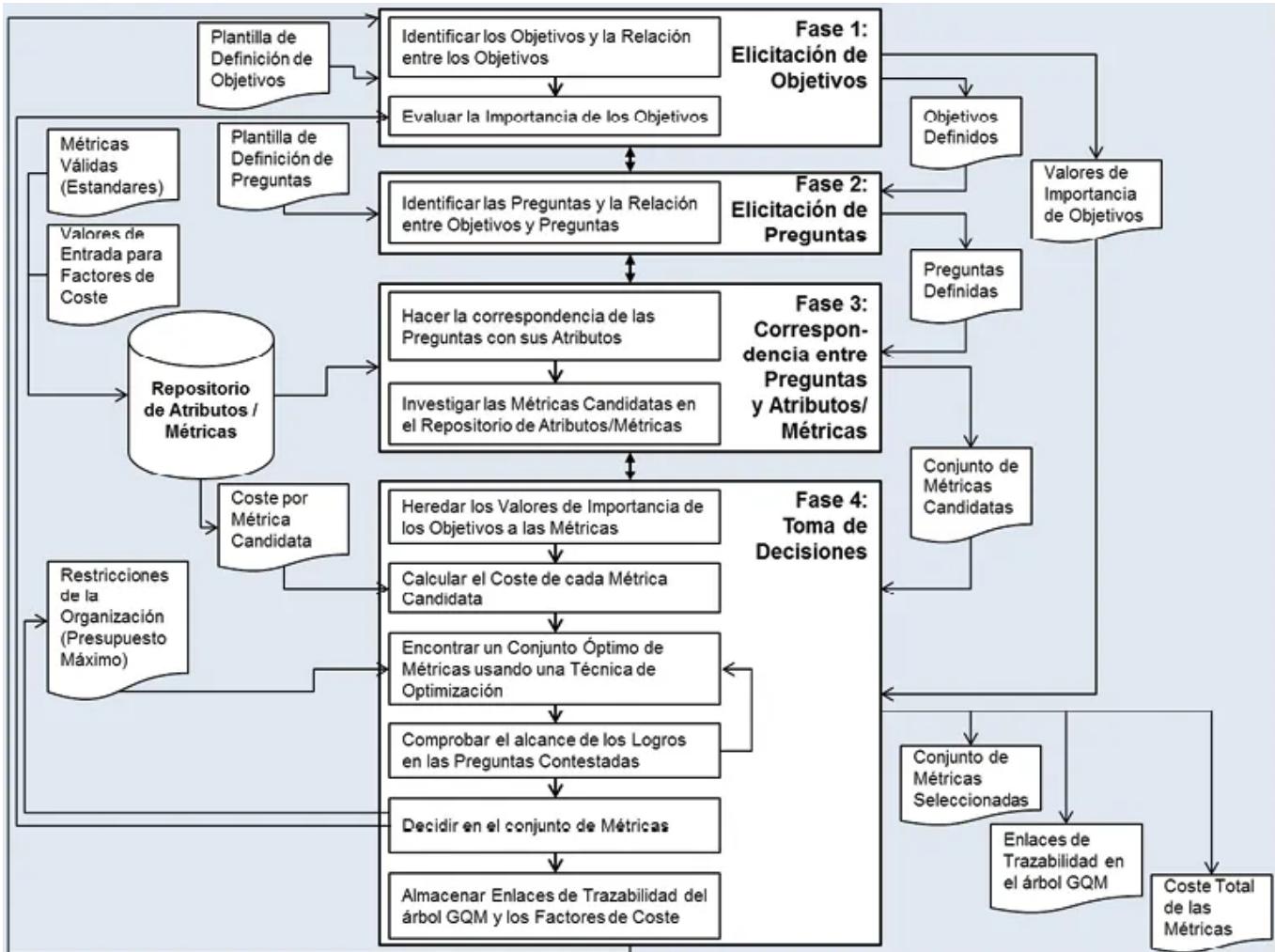


Figura 2.6 Proceso de GQM-DSFMS (Gencel et al., 2013)

Mediante la propuesta GQM-DFMS se trata además de dar soporte a entornos altamente dinámicos donde los objetivos de las organizaciones, sus restricciones, necesidades de información y prioridades cambian.

GQM+ STRATEGIES

GQM+ Strategies es un enfoque para alinear los objetivos y estrategias de negocio de una organización a lo largo de sus distintas unidades o departamentos por medio de la medición (Basili et al., 2014). Para ello se consideran dos perspectivas fundamentales: **Planificación organizacional**, en la que se especifican los objetivos o metas de una organización (*G, goals*) junto con las estrategias para alcanzarlas (*S, strategies*) que establecen las acciones a llevar a cabo; y **Control organizacional**, perspectiva en la que se especifican los controles

adecuados para evaluar la consecución exitosa de los objetivos. Para ello se definen modelos de medición con GQM, de modo que los objetivos organizacionales (G) se asocian con objetivos de medición (MG, *measurement goals*), con preguntas (Q) y métricas (M) que dan soporte a la obtención de información objetiva sobre la satisfacción de los objetivos (ver Figura 2.7).

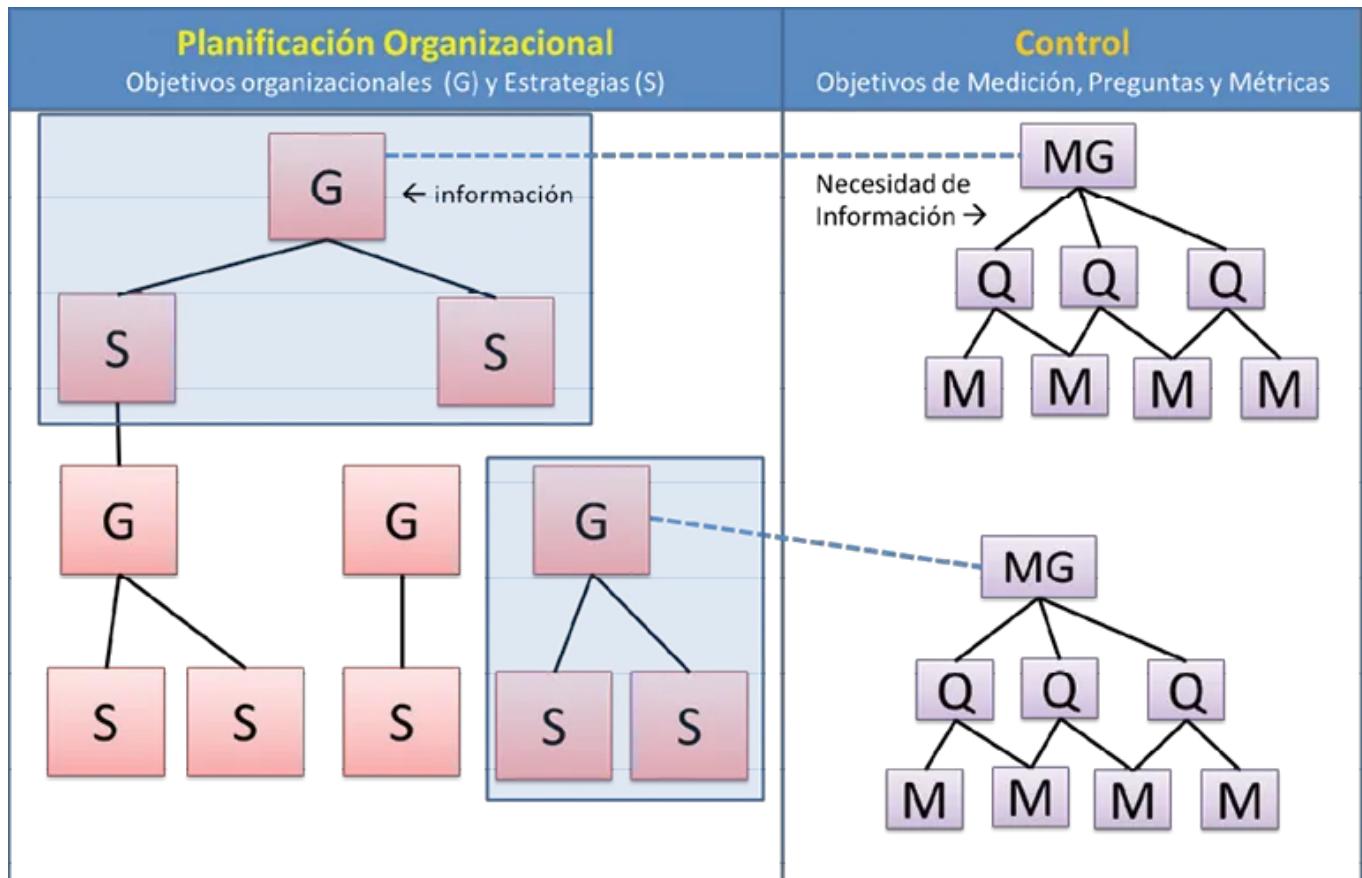


Figura 2.7 GQM+ Strategies: Perspectivas de planificación organizacional y control (Basili et al., 2014)

El proceso a seguir en GQM+ Strategies se muestra en la Figura 2.8, en la que, tal como se puede observar, el proceso se basa en el ciclo clásico PDCA (Plan, Do, Check, Act), y está formado por seis fases que se agrupan en las siguientes etapas:

- **Desarrollar (Develop)**, durante la cual se desarrolla un modelo o cuadrícula (*grid*) que establece el alineamiento entre objetivos, estrategias y los datos de medición necesarios. En la primera fase se

caracteriza la situación actual de la organización a partir de la cual en la fase 2 se define la cuadrícula.

- **Implementar** (*Implement*), que consiste en la ejecución de las estrategias y mediciones definidas en la cuadrícula, para comprobar el logro de dichos objetivos y la efectividad de las estrategias, entre otros aspectos. En la fase 3 se especifican los planes para recopilar los datos de las mediciones y ejecutar las estrategias establecidas en la cuadrícula. En la fase 4 se ejecutan y analizan los planes para comprobar el cumplimiento de los mismos. Si no se cumplen los objetivos, el líder del proyecto de mejora correspondiente puede hacer los ajustes locales necesarios en la cuadrícula en tiempo de ejecución. Si se alcanza un cierto hito o se produce algún disparador (*trigger*), como por ejemplo que un cierto objetivo no se pueda alcanzar sin cambios globales que van más allá del alcance y recursos del proyecto, hay que ir a la fase 5.
- **Aprender** (*Learn*), que implica el aprendizaje a partir del análisis de los resultados y mejora del proceso para la generación de objetivos y estrategias adicionales. En la fase 5 se analiza la consecución de los objetivos y se realiza un análisis de las causas raíz para el éxito o fracaso de las estrategias. En la fase 6 se registra lo aprendido en las fases previas y se solicitan acciones de mejora si los resultados reales difieren de los planificados, y que pueden abarcar desde la revisión de características y suposiciones del contexto a ajustes de la estructura de objetivos y estrategias o la redefinición de la forma en que se realizan o interpretan las mediciones.



Figura 2.8 Proceso de GQM+ Strategies (Basili, Trendowicz et al. 2014)

SINIS

Trinkenreich et al. (2018) desarrollan SINIS (*Goals, Strategies, Indicators for IT Services*) como complemento a GQM+ Strategies. Este método sirve para identificar Objetivos, Estrategias e Indicadores para Servicios de TI, mediante la siguiente serie de actividades con sus correspondientes plantillas:

- Elicitar Servicios de TI, Factores de Contexto y Suposiciones
- Definir Objetivos de los Servicios de TI
- Crear Planes de Medición para satisfacer los objetivos de Servicios de TI
- Crear Modelos de Interpretación para los indicadores relacionados con los objetivos de Servicios de TI
- Analizar Procesos Críticos

- Establecer Estrategias para los procesos críticos
- Crear Planes de Medición para las estrategias
- Crear Modelos de Interpretación para las estrategias
- Construir, revisar y ajustar las Estrategias GQM+

Casos de Aplicación de GQM

A continuación, enumeramos cronológicamente algunos de los ejemplos más relevantes de utilización de GQM. En la Tabla 2.3 se presentan casos con aplicación industrial, en la Tabla 2.4 se muestran otros casos de GQM realizados en el contexto de varios dominios de aplicación, y en la Tabla 2.5 se resumen los principales casos de aplicación de GQM+.

Fuente	Empresa/ Organismo	Objetivo y descripción
Daskalantonakis (1992)	Motorola	Conocer mejor el proceso de desarrollo y ser capaces de tomar decisiones para mejorar la productividad y la calidad. Entre los beneficios objetivos que se obtuvieron destaca el 50% de la reducción de la densidad de defectos en 3,5 años.
Bhandari et al. (1995)	IBM	Se utiliza GQM conjuntamente con una técnica de descubrimiento del conocimiento conocida como “Attribute Focusing” para analizar los datos de la encuesta de satisfacción de usuarios de IBM
Basili et al. (1996)	NASA	Entender y predecir el proceso de mantenimiento de software, en desarrollos de la NASA para una serie de satélites científicos y los relativos al Space Shuttle.
Kontio et al. (1996)	NASA	Se aplica GQM para definir criterios de evaluación para componentes reutilizables de software, en el marco del método OTSO (<i>Off-The-Shelf Option</i>).

		Algunos de los atributos de evaluación son: memoria utilizada por el componente cuando se carga, tiempo para inicializarlo, número de errores encontrados durante la evaluación, estimación del crecimiento potencial del vendedor, apariencia de la interfaz, etc.
Birk et al. (1998)	Schlumberger Retail Petroleum Systems	Se presentan los resultados de aplicación de GQM durante 4 años, señalando sus costes y beneficios
Fuggetta et al. (1998)	Digital Software Engineering Center de Gallarate (Italia).	Se aplica GQM para la mejora de la fiabilidad y la reusabilidad de software. Se reportan además como beneficios adicionales para la empresa: mejores prácticas en la recogida de datos, mejores sistemas de gestión de datos, mejor interpretación de datos, mejor motivación en la recogida de datos y mejor utilización de datos existentes.

Solingen y Berghout (1999)	4 casos de estudio industriales	El primer caso de estudio es un programa de medición cuyos objetivos consistían en analizar, en un proyecto de desarrollo software, tanto el producto como el proceso para entender su fiabilidad y sus causas, así como la efectividad de la reutilización. La aplicación del programa de medición resultó un éxito y se obtuvieron como principales resultados un mejor entendimiento del producto y del proceso, así como el incremento de experiencia en la realización de programas de medición.
----------------------------	---------------------------------	---

		<p>En el segundo caso se presenta el programa de medición “Reviews and Inspections”, centrado en investigar los efectos de las revisiones (por pares: desarrollador y el inspector) en la detección de fallos. El proyecto analizado fue el desarrollo de un sistema de gestión de estaciones de servicio, de seis años de duración y cuyo producto estaba escrito mayoritariamente en C++.</p>
		<p>El tercer caso de estudio consistía en investigar las causas y efectos de las interrupciones en el trabajo de los ingenieros software. El equipo del proyecto (dos ingenieros software, dos hardware y el jefe del proyecto) se dedicaba al desarrollo y mantenimiento de la unidad central de control de un dispensador de combustible en una estación de servicio. Estos datos sirvieron en la empresa para implementar mejoras, por ejemplo, reduciendo el número de interrupciones relacionadas con aspectos de documentación en un 80% mediante una actualización y mejor distribución de los documentos.</p>
		<p>El cuarto caso de estudio se centró en analizar la carga de trabajo de un equipo de proyecto que tenía que reorganizarse al cambiar su perfil de trabajo de desarrollo a mantenimiento.</p>
Lavazza (2000)	Pirelli Cavi (división italiana de la empresa Pirelli Cables and Systems)	<p>Se resume la experiencia del programa de medición de la empresa para la extracción de datos conforme a procesos y fiables en el desarrollo de un producto de una red de gestión de telecomunicaciones.</p> <p>Para automatizar el proceso de medición se construyó una herramienta denominada GQM Tool.</p>
Morisio et al.	NASA	<p>Se utiliza GQM para mejorar el desarrollo de</p>

(2000)		software basado en componentes COTS
Lindvall et al. (2002)	Instituto Fraunhofer de EEUU	Se utiliza GQM para demostrar la mejora producida en la reestructuración de una arquitectura de software
Aversano et al. (2004)	Departamento de la Administración Pública de Benevento (Italia)	Se aplica GQM en un proyecto de reingeniería de procesos de negocio (BPR), creando un marco de medición que ayudó a los analistas de los procesos de negocio a encontrar indicaciones útiles relativas al rendimiento del proceso, sus elementos críticos, salidas a mejorar, nuevos servicios requeridos, el impacto de los cambios escogidos, mejoras esperadas y herramientas de soporte.
Manhart y Schneider (2004)	Daimler-Chrysler	Se aplica GQM para el desarrollo ágil de software embebido con dos objetivos de negocio: incrementar el total de funciones basadas en software específicas de usuario (en este caso relacionadas con los vehículos) y lograr cero defectos en los procesos de software.
Berander y Jönsson (2006)	Ericsson AB (Suecia)	Se resume la aplicación de la versión extendida de GQM que proponen para evaluar los procesos de gestión de cambios e ingeniería de requisitos.
Lavazza y Mauri (2006)	Banca Cobota	Se define un programa de medición en un pequeño banco.
Vieira et al. (2011)	Critical Software	Explican cómo se ha utilizado GQM en la empresa Critical Software para promover la reutilización del software.
Tarhan y Yilmaz (2014)	Empresa de telecomunicaciones	Explican los resultados de un análisis sobre el desempeño del proceso de desarrollo y la calidad del producto de sus procesos incrementales y ágiles aplicados en una empresa de desarrollo de software de telecomunicaciones con 65 empleados. La aplicación de GQM ayudó a la hora de proporcionar objetivos adecuados y las medidas y preguntas relacionadas, permitiendo la definición de un programa de medición

		ordenado y que facilitaba la interpretación de los resultados.
Ram et al. (2018)	4 empresas	Se utiliza GQM para investigar las causas que explican la elección de métricas en desarrollos ágiles de software.

Tabla 2.3. Casos de Aplicación de QGM

Fuente	Dominio de Aplicación	Descripción
Benedicenti et al. (1996)	Reutilización Software	Aplicación de GQM en un programa que pretende institucionalizar un proceso con actividades específicas de reutilización de software en desarrollo orientado a objetos.
Houdek y Kempter (1997)	Experimentación	Se usa GQM como manera de aprender y capturar experiencias al analizar los datos obtenidos en experimentos, que junto con otro tipo de información constituyen “paquetes de experiencias”.
MacDonell y Fletcher (1998)	Multimedia	Aplicación de GQM para caracterizar los sistemas multimedia, especialmente para determinar el esfuerzo requerido para su desarrollo.
Bonifati et al. (2001)	Almacenes de Datos	Se utiliza para el diseño de almacenes de datos, representando los requisitos del almacén como objetivos GQM.
Basili et al. (2002)	Laboratorio de Ingeniería del Software	Se repasan los 25 años del Software Engineering Laboratory, y la evolución en el esfuerzo realizado por mejorar el software mediante la medición y, especialmente, la aplicación de GQM.
Geppert y Weiss (2003)	Líneas de Producto	Utilizan GQM para desarrollar un programa de medición y determinar qué medidas serían usadas para detectar los dominios candidatos en un entorno de líneas de producto.
Scholtz y Steves (2004)	Sistemas CSCW (Computer-Supported	Se usa GQM para evaluar sistemas CSCW, siendo uno de los objetivos “que la herramienta de mensajes genérica proporcione comunicación síncrona y

	Cooperative Work)	asíncrona”, mientras que las métricas se basaban en la efectividad, la satisfacción del usuario de la interfaz, entre otras.
Von Konsky y Robey (2005)	Entorno académico	Presentan un caso de estudio que combina un subconjunto de TSP (Team Software Process) con una serie de métricas identificadas utilizando GQM, que fueron aplicadas para evaluar la efectividad de los roles de gestión de proyectos asignados a cada miembro del equipo de proyecto. Se demuestra que el enfoque utilizado puede servir para evaluar el aprendizaje de los estudiantes y enfatizar la significación de la gestión de proyectos software.
Hussain y Ferneley (2008)	Aplicaciones móviles	Se aplica GQM para desarrollar una métrica de usabilidad para aplicaciones móviles.
Montini et al. (2009)	Sistemas en Tiempo Real	Pruebas de patrones de diseño de sistemas de tiempo real y empotrados.
Al-Nanih et al. (2009)	Interacción Persona-Ordenador	Construcción de un modelo de calidad en uso, aplicable a los sistemas de interacción persona-ordenador.
Khomh et al. (2011)	Patrones Software	Usan GQM para construir redes bayesianas para la detección de antipatrones.
Monden et al. (2012)	Proyectos Software	Adapitan los modelos GQM para la monitorización de proyectos software.
Yahaya et al. (2013)	Mantenimiento del Software	Utilizan GQM para la medición y la clasificación del “envejecimiento” (<i>ageing</i>) del software.
Aslan et al. (2014)	Procesos-Productos Software	Utilizan GQM para estudiar cómo los datos de realización (<i>enactment</i>) del proceso afectan a la predicción de defectos de los productos software.
Carvalho et al. (2015)	Sistemas Ubiuos	Aplican GQM para la evaluación de la característica “calmness” en sistemas ubicos, de modo que en la interacción con los usuarios el sistema no debe ser interrumpida de forma innecesaria. Como resultado definen su modelo de medición usando GQM y lo aplican en un caso de estudio evaluando tres

		aplicaciones móviles.
Lavazza et al. (2015)	Procesos de Medición	Aplican un proceso para la definición de indicadores clave de rendimiento (KPI) e indicadores de éxito, que incluye GQM para la definición de las métricas y el lenguaje estadístico R para el cálculo de los indicadores y su análisis. Como resultado del caso de estudio se concluye que el proceso de definición y evaluación de indicadores propuesto es adecuado para la definición e implementación de los indicadores.
Luo y van den Brand (2016)	Seguridad física	Proponen métricas para evaluar la seguridad física y para ello proponen una metodología para diseñar este tipo de métricas desde diferentes dimensiones, incluyendo la realización de encuestas aplicando GQM, junto con la aplicación de un procedimiento basado en PSM y tormenta de ideas. Como resultado llevan a cabo un caso de estudio obteniendo 76 métricas y validando las mismas con una encuesta distribuida a 24 expertos.
Yahya et al. (2017)	Seguridad en Cloud Computing	Se aplica GQM para el desarrollo de un modelo de evaluación de la seguridad en un entorno cloud basado en necesidades reales.

Tabla 2.4. Otros casos de aplicación de GQM

Fuente	Caso y descripción
Sarcia' (2010)	Se aplica una variante de GQM+ a un dominio de desarrollo no software, como el entrenamiento militar.
Heidrich y Trendowicz (2011)	Se usa GQM+ en el “Software Engineering Center de la Information-technology Promotion Agency” de Japón, destacando como factores de éxito para alinear los proyectos con la estrategia organizacional y evaluar además el riesgo del no alineamiento, la documentación transparente de objetivos y estrategias y la recolección de KPIs.
Munch et al. (2013)	Presentan la aplicación durante cinco meses de GQM+Strategies para elicitar, enlazar y alinear los objetivos de una organización de desarrollo de productos de sistemas a lo largo de varios niveles organizacionales.

Basili et al. (2013)	Describen cómo la empresa ECOPETROL está utilizando GQM+Strategies para medir el alineamiento entre el negocio y las TI con el fin de mejorar su competitividad.
Basili et al. (2014)	Se describen ejemplos de aplicación de GQM+Strategies en diferentes organizaciones de desarrollo de software.
Petersen et al. (2014)	Explican cómo han desarrollado en Ericsson AB (Suecia) un instrumento denominado GQM+S-EI (GQM+Strategies Elicitation Instrument) que permite eliciar la información de los stakeholders de manera precisa y completa.
Lopez et al. (2016)	Aplican GQM+Strategies en una empresa grande multi-sectorial, obteniendo importantes beneficios como mejores alineamiento e integración de los diferentes objetivos y una visión holística de los mismos. Asimismo la aplicación de GQM+Strategies permitió eliminar redundancias y esfuerzos duplicados en la empresa al proporcionar un mejor equilibrio entre los objetivos y las estrategias, concluyendo que la aplicación de este método en contextos de empresas grandes y complejas es adecuado, aunque requiere un importante esfuerzo.
Mandic y Gvozdenovic (2017)	Extienden GQM+ Strategies para incluir capacidades para evaluar la relación de objetivos y estrategias organizacionales mediante el análisis causal.
Trinkenreich et al. (2017)	Se utiliza GQM+ Strategies para establecer estrategias para conseguir objetivos de servicios de TI, señalando cómo ayuda a que se monitoricen mejor los resultados de las acciones y su alineamiento con los objetivos.

Tabla 2.5. Casos de Aplicación de QGM+ Strategies

Por último, en la tabla 2.6. se recogen los factores críticos de éxito (FCE) en la aplicación de GQM (Birk, et al., 1998).

Factores de éxito generales

- Compromiso de la dirección
- Nominación y disponibilidad de un patrocinador
- Actitud positiva del proyecto y la organización hacia la medición
- Identificación de objetivos y acciones de mejora
- Planificación detallada del programa de medición
- Existencia de un modelo descriptivo de los procesos a medir

- Acceso a investigadores y resultados de investigación
- Medición y mejora de los procesos de medición
- Definición explícita y operacional de los procesos de medición

Factores de éxito para la aplicación inicial

- Coordinación del programa de medición con la planificación del proyecto
- Número pequeño de objetivos
- Contemplación de los puntos de vista de todos los roles importantes del proyecto
- Objetivos de medición que sean fácilmente mesurables e interpretables
- Soporte de herramientas completo y sencillo
- **Entrenamiento específico de roles**

Factores de éxito para la aplicación rutinaria

- Soporte de herramientas avanzadas

Tabla 2.6. FCE en la aplicación de GQM (Birk, et al., 1998)

LECTURAS RECOMENDADAS

- *Basili, V., Trendowicz, A., Kowalczyk, M., Heidrich, J., Seaman, C., Münch, J., Rombach, D. (2014). Aligning Organizations Through Measurement. The GQM+ Strategies approach. Springer.*

En este libro se presenta la estrategia GQM+ que permite alinear los objetivos de negocio con los tecnológicos, mediante una extensión de GQM.

SITIOS WEB RECOMENDADOS

- www.computer.org

Sitio web de la IEEE Computer Society, donde se pueden encontrar tanto revistas como conferencias en las que aparecen publicados con frecuencia trabajos utilizando o ampliando GQM.

3

ESTÁNDARES DE MEDICIÓN

INTRODUCCIÓN

Como soporte al proceso de medición se pueden destacar diversos marcos de trabajo como PSM (*Practical Software Measurement*), así como ciertos estándares, entre los que destacan ISO 15939.

El objetivo de estos estándares y marcos de trabajo es proporcionar la referencia necesaria para poder llevar a cabo el proceso de medición de una forma efectiva y sistemática, en base a una serie de objetivos.

En este capítulo resumimos estos marcos y estándares relacionados con la medición.

PRACTICAL SOFTWARE MEASUREMENT (PSM)

Descripción

La metodología PSM (*Practical Software Measurement*) (Basili et al., 2002), patrocinada por el Departamento de Defensa de EEUU, se basa en la experiencia adquirida por docenas de organizaciones para saber cuál es la mejor manera de implementar un programa de medida de software con garantías de éxito. No se trata de una aproximación genérica, sino que incluye líneas guía para ajustar los marcos de trabajo de la medida y las prácticas a la situación de cada proyecto en cada organización.

PSM propone un “Modelo de Procesos de Medición” que se divide en cuatro actividades principales:

- **Planificación de la medición.** En esta actividad se definen las métricas que proporcionen la suficiente visibilidad en los proyectos para satisfacer las necesidades de información que PSM agrupa en siete categorías (véase Tabla 3.1). Esta actividad incluye identificar qué necesitan saber los beneficiarios de la medición (encargados de la toma de decisiones), relacionar las necesidades de información con las entidades que pueden ser medidas, y seleccionar y especificar métricas basadas en los proyectos y en los procesos organizacionales.
- **Realización de la medición.** Esta actividad implica reunir los datos de las mediciones, realizar el análisis y presentar los resultados para que la información pueda ser útil para la toma de decisiones.
- **Evaluación de la medición.** En esta actividad tanto el proceso de medición como las propias métricas definidas deben evaluarse y mejorarse periódicamente según sea necesario.
- **Establecimiento y mantenimiento del compromiso.** Esta actividad implica establecer los recursos, formación y herramientas necesarias para implementar un programa de medición de forma efectiva y, lo que es más importante, asegurar que la dirección usa la información producida.

Planificación y progreso
Recursos y costes
Tamaño y estabilidad del producto
Calidad del producto
Desempeño del proceso
Efectividad de la tecnología
Satisfacción del cliente

Tabla 3.1. Categorías de necesidades de información según PSM

Un aspecto básico para lograr programas de medición efectivos es el hecho de disponer, al final del proceso, de información útil para los encargados de la toma de decisiones. Para ello PSM incorpora un modelo de información de la medición que relaciona las entidades que son medidas con las medidas definidas y en última instancia con las necesidades de información que se satisfacen. Este modelo incorpora una estructura denominada constructor de la medición que describe cómo los atributos relevantes de los productos y procesos se cuantifican y se convierten en indicadores, que son elementos que proporcionan la base necesaria para la toma de decisiones. Todo constructor de la medición implica tres niveles de medida: medidas base, medidas derivadas e indicadores.

En definitiva, PSM proporciona un método sistemático para la planificación y realización del proceso de medición y análisis, y constituye la base del estándar internacional ISO/IEC/IEEE 15939.

Casos de Aplicación de PSM

En cuanto a la aplicación de PSM, a continuación, se resume un conjunto de propuestas representativas:

- En la propia guía de PSM (DoD, 2003) se pueden encontrar ejemplos de aplicación de la metodología a tres proyectos de desarrollo de software del departamento de defensa de los Estados Unidos.
 - El primer caso de estudio consiste en la aplicación de la medición sobre un proyecto de desarrollo de un sistema sofisticado de armamento para barcos de combate. El enfoque de desarrollo se basa en la mejora de un sistema existente utilizando componentes COTS (*commercial off-the-shelf*) y reutilizando software en una arquitectura renovada y revisada. En este caso de estudio se ilustra cómo la implantación del programa de medición ayudó a identificar y analizar de forma

objetiva los diferentes aspectos en el desarrollo del producto software y cómo el jefe del proyecto usó la información resultante para la toma de decisiones.

- Un sistema de información para la gestión del personal militar constituye el contexto del segundo caso de estudio, en el que se ilustra la aplicación de la metodología sobre un proyecto de desarrollo en curso, como consecuencia de no alcanzar satisfactoriamente uno de los hitos del proyecto respecto a calendario y costes. El proceso de medición fue adaptado a las características especiales del sistema, utilizando métricas tales como puntos función, y definiendo nuevas métricas para dar soporte al proceso de instalación. La aplicación de la metodología permitió recuperar el control del proyecto y los miembros del equipo del proyecto fueron conscientes de la importancia de la medición para identificar y resolver tanto aspectos técnicos como de gestión.
- El escenario del tercer y último caso de estudio es el desarrollo de un software de soporte para radares del ejército. Este caso ilustra el uso de la medición en un proyecto que ha estado operativo durante treinta años y que se aplica en las tareas de mantenimiento mediante la estimación del coste e impacto de las peticiones individuales de cambio.
- En Ishigaki y Jones (2003) se presenta la aplicación de PSM al Proceso Unificado de Rational (RUP). En la Tabla 3.2 se presenta la correspondencia entre las categorías de información de PSM y las fases de RUP.

Fase de RUP Categoría de información

Inicio	Planificación y progreso Recursos y costes Desempeño del proceso
--------	--

Elaboración	Planificación y progreso Recursos y costes Tamaño y estabilidad del producto Calidad del producto Desempeño del proceso Efectividad de la tecnología
Construcción	Planificación y progreso Recursos y costes Tamaño y estabilidad del producto Calidad del producto Desempeño del proceso Efectividad de la tecnología Satisfacción del cliente
Transición	Planificación y progreso Recursos y costes Calidad del producto Satisfacción del cliente

Tabla 3.2. Correspondencia entre PSM y RUP (Ishigaki y Jones, 2003)

- Murdoch et al. (2003) discuten la aplicación de PSM al dominio de seguridad de funcionamiento (*safety*). Además de los conceptos que aparecen en la guía de PSM, los autores identificaron tres conceptos nuevos: *dependability-safety*, *assurance-safety*, y *scope-safety*; así como diversas medidas: el número de riesgos, el número de modo de fallos, el número de escenarios de fallos, el número de incidentes, etc.
- Card (2003) explica cómo integrar PSM y el cuadro de mando integral (*balanced scorecard*) ya que la perspectiva financiera del cuadro de mando integral se relaciona con la categoría de recursos y costes del PSM, la de proceso interno con la categoría de desempeño del proceso, y la perspectiva del cliente con la categoría de satisfacción del cliente.

ISO/IEC/IEEE 15939

Descripción

Este estándar internacional (ISO, 2017b), basado en PSM, identifica las actividades y tareas necesarias para identificar, definir, seleccionar, aplicar y mejorar la medición de software dentro de un proyecto o de la estructura de medición de una organización.

De acuerdo a este estándar, el principal propósito del proceso de medición es recoger, analizar y reportar información y datos objetivos para soportar la gestión efectiva y demostrar la calidad de los productos, servicios y procesos. Como resultado de implementar el proceso de medición:

- Se identifican las necesidades de información
- Se identifica o desarrolla un conjunto apropiado de medidas basadas en las necesidades de información
- Se recogen, verifican y almacenan los datos necesarios
- Se analizan los datos y se interpretan los resultados
- Los elementos de información proporcionan información objetiva para soportar la toma de decisiones
- Se mantiene el compromiso organizacional con la medición
- Se planifican las actividades de medición identificadas
- Se evalúan los procesos de medición y las medidas
- Se comunican las mejoras al propietario del proceso de medición

El proceso de medición de software propuesto en el estándar se compone de cuatro actividades principales (Figura 3.1) que se suceden en un proceso iterativo permitiendo una realimentación y una mejora continua del proceso de medición. Los *Procesos Técnicos y de Gestión* de una organización no se encuentran dentro del ámbito de este estándar, aunque son una interfaz

externa importante para las actividades de medición que se incluyen en el estándar.

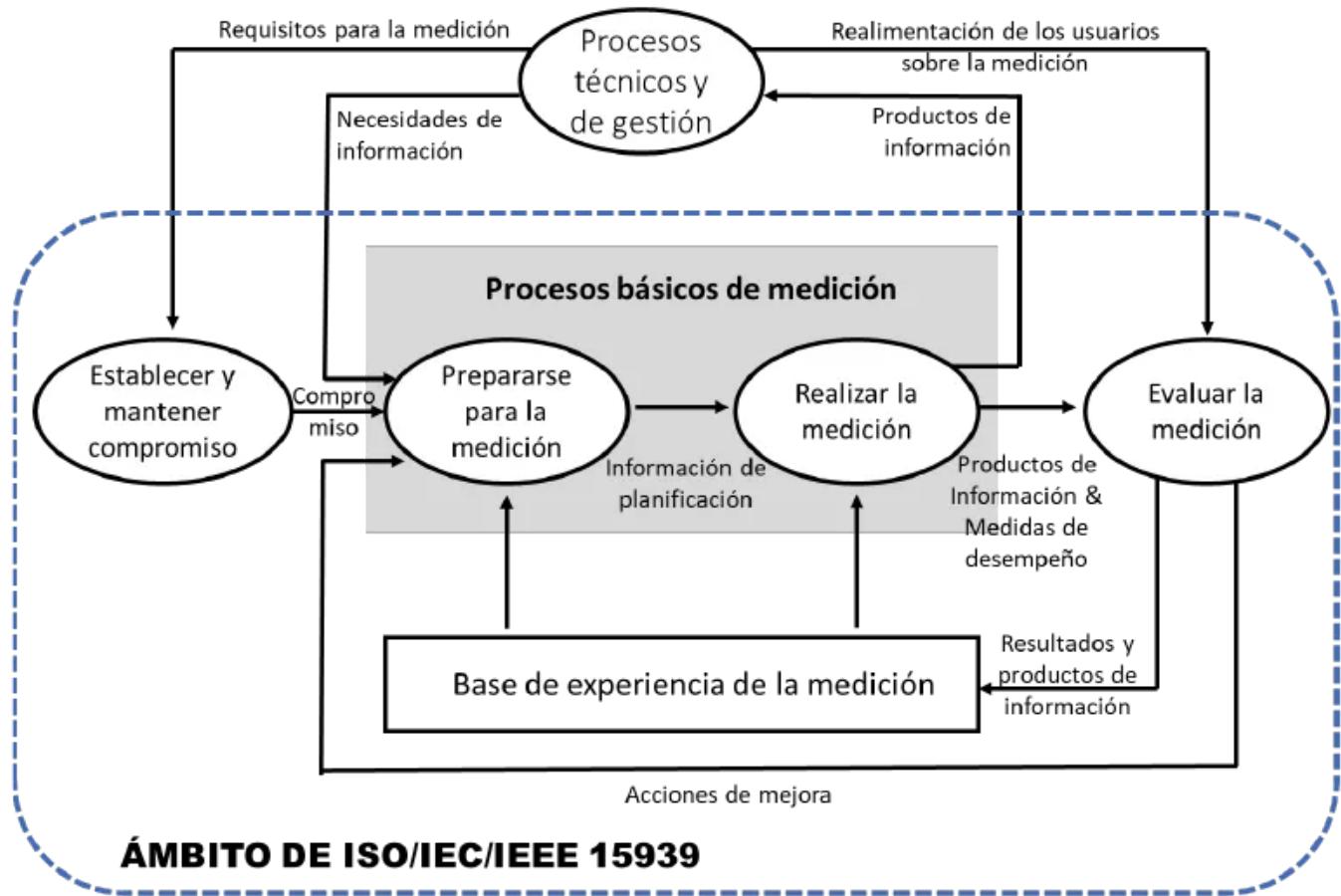


Figura 3.1 Modelo de Procesos de Medición de Software (ISO/IEC, 2017)

Las tareas para las diferentes actividades de los procesos de medición de acuerdo al estándar son:

1. Establecer y mantener el compromiso

Este proceso se compone de las siguientes actividades:

- **Aceptar los requisitos para la medición**, que incluye como tareas: identificar el alcance de la medición (proyecto, área funcional, organización, etc.; así como los stakeholders), establecer el compromiso de la dirección, y comunicar el compromiso a la unidad organizacional.
- **Asignar recursos**, lo que conlleva: asignar la responsabilidad sobre la medición, y proporcionar recursos para planificar,

realizar y evaluar la medición.

2. Prepararse para la medición

Este proceso se compone de las siguientes actividades:

- Definir la estrategia de medición, incluyendo los roles, responsabilidades, rendiciones de cuenta y autoridades; las actividades para proveedores o subcontratistas, los flujos de información, los recursos y la integración con otros procesos.
- Describir las características de la organización que son relevantes para la medición.
- Identificar y priorizar las necesidades de información, lo que conlleva además: seleccionar las necesidades de información que se van a abordar y registrar y comunicar las necesidades de información seleccionadas.
- Seleccionar y especificar las medidas que satisfacen las necesidades de información, lo que incluye: identificar medidas candidatas que satisfagan las necesidades de información seleccionadas, seleccionar las medidas a partir de las medidas candidatas, y especificar las medidas seleccionadas.
- Definir los procedimientos para la recogida, análisis, acceso y reporte de los datos, lo que conlleva también definir los procedimientos de gestión de la configuración.
- Definir criterios para evaluar los elementos de información y los procesos de medición.
- Identificar y planificar los sistemas o servicios habilitadores que se usarán.
- Revisar, aprobar, y proporcionar recursos para las tareas de medición.
- Adquirir y desplegar las tecnologías de soporte, lo que incluye la evaluación y selección de las tecnologías apropiadas.

3. Realizar la medición

Este proceso se compone de las siguientes actividades:

- Integrar los procedimientos para la generación, recogida, análisis y reporte de datos en los procesos relevantes, y por supuesto, comunicar estos procedimientos de datos a los proveedores de los datos y a los *stakeholders*.
- Recoger, almacenar y verificar los datos, incluyendo la información de contexto.
- Analizar los datos y desarrollar los elementos de información, revisando éstos para comprobar que se han interpretado correctamente y que las necesidades de información se han satisfecho.
- Registrar los resultados e informar a los usuarios de la medición.

4. Evaluar la medición

Este proceso se compone de las siguientes actividades:

- Evaluar los productos de información y el proceso de medición, respecto a los criterios de evaluación especificados; así como almacenar las lecciones aprendidas en la base de experiencia de la medición.
- Identificar mejoras potenciales, tanto a los productos de información como a la medición, y comunicarlas.

Casos de aplicación de ISO/IEC/IEEE 15939

En la Tabla 3.3 se resumen por orden cronológico una serie de casos representativos de aplicación de ISO/IEC 15939.

Fuente	Descripción
De Oliveira y Belchior (2006)	Se presenta un proceso de medición de software resultante del mapeo de los procesos de medición de CMMI-SW, ISO/IEC/IEEE 15939, IEEE 1061, Seis sigma y PSM.

Staron et al. (2008)	Se aplica para crear un marco para diseñar sistemas de medición y su evaluación en proyectos reales en Ericsson, que permite incrementar la eficiencia de la recogida de métricas y la adopción de las métricas en la organización.
Caballero et al. (2009)	Se propone una metodología basada en ISO/IEC/IEEE 15939 para desarrollar planes de medición de calidad de datos.
Tsunoda et al. (2010)	Se propone un modelo para la monitorización de proyectos con stakeholders basado en el estándar y añadiendo los objetivos de los stakeholders, los KPIs, las acciones correctivas y el tiempo de verificación.
Staron et al. (2010)	Se presenta un proceso industrial, aplicado en Ericsson durante más de tres años, para desarrollar sistemas de medición incluyendo los artefactos y entregables importantes para desplegar con éxito el sistema de medición en la industria.
Staron (2012)	Se investiga en base a la ISO/IEC/IEEE 15939 cómo utilizar medidas de una manera efectiva en el proceso de toma de decisiones, aplicándolo en un estudio de caso en Ericsson.
Kurtel (2013)	Se presenta un proceso para la evaluación, desempeño y planificación de la medición del mantenimiento en una empresa de desarrollo de software de Turquía, combinando ISO/IEC/IEEE 15939 con ISO/IEC 9126 e ISO/IEC 14598.
Vianden et al. (2013)	Se propone una arquitectura para una infraestructura de medición organizacional federal basada en mediciones orientadas a servicio, siguiendo el flujo de datos de ISO/IEC/IEEE 15939
Staron et al. (2013)	Se propone un método para evaluar la completitud de la información proporcionada por los sistemas de medición y su aplicación en la empresa Ericsson, donde se aplicó para proporcionar a los stakeholders un sistema temprano de alertas sobre problemas con la calidad de software.
Feyh y Petersen (2013)	Se aplica para identificar medidas e indicadores propuestos en la bibliografía sobre desarrollo de software <i>lean</i> .
Kurtel y Ozemre (2013)	Se usa para definir un marco para la planificación de la medición y aplicarlo en una empresa de desarrollo de software en el sector logístico en Turquía.
Assila et al. (2014)	Se propone la integración de datos de evaluaciones cuantitativa y

	cualitativa para interfaces persona-ordenador.
Staron et al. (2014)	Se presentan cuadros de mando para monitorizar el progreso del desarrollo de software en las empresas Ericsson, Volvo Car Corporation, y Saab Electronic Defense Systems.
Staron y Meding (2015)	Se alinea con el estándar el concepto de “medición como servicio” (Measurement-as-a-Service), con el fin de optimizar los procesos de medición en empresas modernas utilizando computación en la nube.
Pavapootanont y Prompoon (2015)	Se utiliza para definir una métrica de usabilidad para un juego móvil en la fase de diseño, a partir de atributos relativos al contenido, dispositivo y juego.
Rana y Staron (2015)	Se utiliza el modelo de información de la medición de la norma para proporcionar un marco general que permita aplicar aprendizaje automático para la evaluación y predicción de la calidad del software.
Assila et al. (2016)	Se utiliza la norma para la integración de datos generados por diferentes métodos y herramientas para la evaluación de la usabilidad.
Mellegård (2017)	Se utiliza la norma para definir perfiles de <i>backlogs</i> de defectos para grandes organizaciones que desarrollan productos complejos (p.ej. fabricantes de automóviles).
Assila et al. (2017)	Se propone un indicador para medir el desempeño de la navegación en entornos virtuales basado en ISO/IEC/IEEE 15939, proporcionando también un método para visualizar los datos.
Tarhan y Altunel (2017)	Se describe un conjunto de necesidades de información y métricas diseñados para medir el efecto de la transformación ágil en una organización software de tamaño mediano.

Tabla 3.3. Casos de aplicación de ISO/IEC/IEEE 15939

LECTURAS RECOMENDADAS

- *Calidad de Sistemas de Información, 5^a edición, Piattini, M., García, F. Pino, F. y García, I., 2019, Ra-Ma, Madrid*, que aporta una visión general sobre la calidad y permite encuadrar el proceso de medición con el resto de procesos del ciclo de vida, los modelos de madurez y la calidad de los procesos, productos, proyectos y servicios.

SITIOS WEB RECOMENDADOS

- www.psmsc.com

Sitio web oficial de la organización PSM, cuyos miembros son expertos en el campo de la medición software, y en el que se puede encontrar la guía de PSM y toda su documentación y material auxiliar.

- www.iso.ch

Sitio Web oficial de la organización ISO donde se puede encontrar el estándar ISO/IEC/IEEE 15939 como estándar de referencia de la medición del software.

4

<https://dogramcode.com/programacion>

MÉTRICAS DE SOFTWARE

INTRODUCCIÓN

Tal y como se ha descrito en los capítulos anteriores, el objetivo de todo proceso de medición es recopilar indicadores sobre entidades software, siendo una entidad software todo elemento software sobre el que se puede aplicar un proceso de medición y que están caracterizadas por una serie de atributos (tamaño, tiempo, etc.). Para realizar la medición es necesario identificar tanto las entidades como los atributos a medir.

Con todo ello, para el estudio de la medición del software hay que estudiar las entidades que pueden ser objeto de medición, así como los atributos característicos de dichas entidades. De acuerdo a modelos de evaluación y mejora como ISO/IEC 33000 o CMMI, a la hora de incrementar el nivel de madurez de una organización hay que establecer una base cuantitativa que de menor a mayor grado de madurez está enfocada sobre:

- **Medición del proyecto**, basado en la gestión de proyectos.
- **Medición del producto**, centrado en su calidad y aspectos técnicos.
- **Medición del proceso**, basado en el estudio y control de la capacidad de los procesos, así como en la gestión de los cambios en el proceso.

La relación entre las métricas de proceso, proyecto y producto se muestra en la Figura 4.1. Como se puede observar en dicha figura, el proceso software constituye la base a partir de la cual se realiza el trabajo dentro de una organización. Dichos procesos se aplican en la práctica en forma de proyectos. Como resultado de la ejecución de proyectos concretos se utilizan recursos y se obtienen productos. Por lo tanto, para establecer un marco de medición dentro de una organización es necesario definir, recoger y analizar métricas sobre el proceso, el proyecto y recursos asociados, así como el producto software.

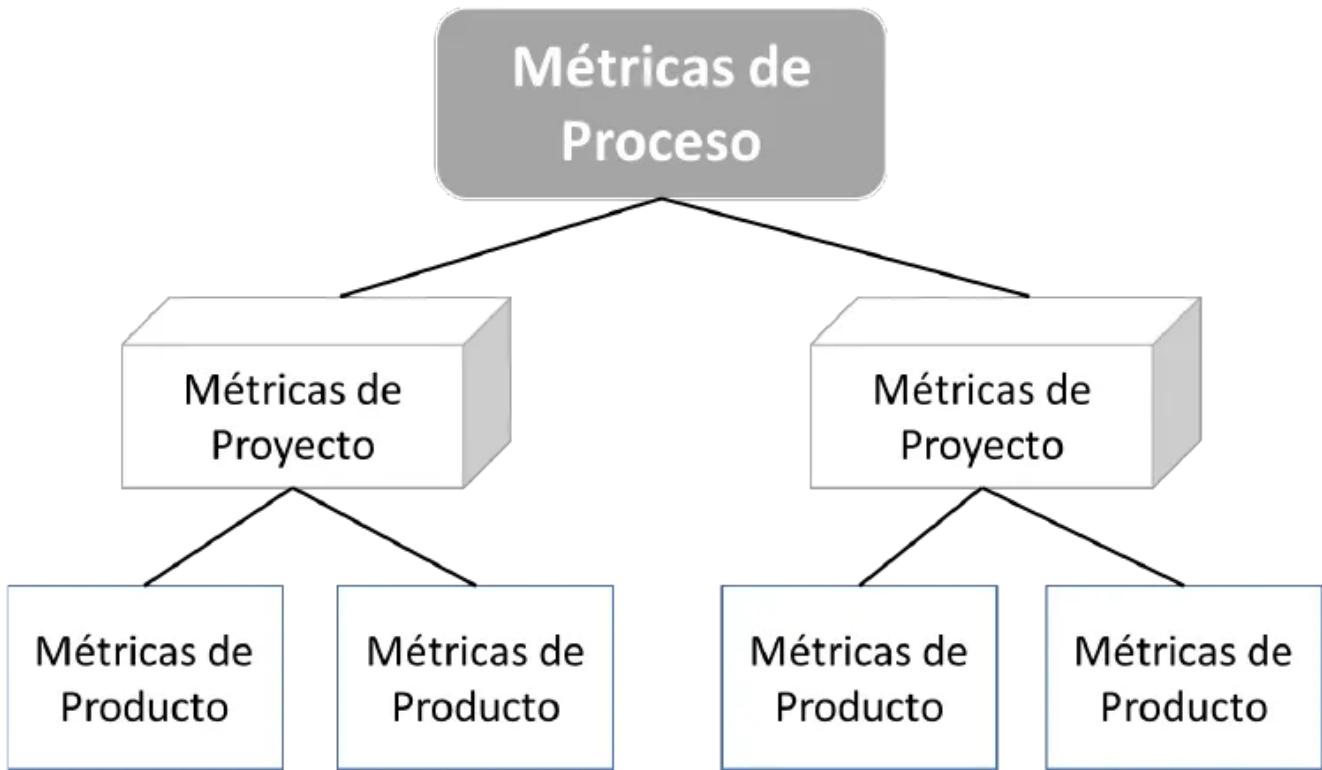


Figura 4.1 Tipos de entidades de medición del software

En este capítulo se describen un conjunto de métricas significativas que pueden ser de gran utilidad en la ayuda a la toma de decisiones para promover la mejora de la calidad y productividad en el desarrollo y mantenimiento del software, clasificadas de acuerdo al tipo de entidad al que pertenecen (proceso, producto, proyecto).

MEDICIÓN DEL PROCESO

La medición del proceso implica las mediciones de las actividades relacionadas con el software siendo algunos de sus atributos típicos el esfuerzo, el coste y los defectos encontrados.

De acuerdo a Pressman (2001) las métricas del proceso de software se utilizan para propósitos estratégicos y, en muchas propuestas, la medición del proceso se realiza extrayendo las características de tareas específicas de la ingeniería de software y obteniendo como resultados

métricas sobre los errores detectados antes de la entrega del software, defectos detectados e informados por los usuarios finales, productos de trabajo entregados, el esfuerzo humano y tiempo consumido, ajuste con la planificación, etc.

Por ello, en la bibliografía, el enfoque de medición del proceso se ha centrado en recopilar una serie de métricas de todos los proyectos y durante un largo periodo de tiempo con el objetivo de proporcionar indicadores que lleven a mejoras de los procesos software a largo plazo. En este sentido, un área clave de investigación es el control estadístico de procesos (Florac y Carleton 1999), tema del que se ofrece una visión más completa en el siguiente capítulo.

En Meidan et al. (2018) se ofrece un mapeo sistemático sobre las diferentes propuestas de medición del proceso software, y los diferentes atributos que se intentan medir: capacidad, madurez, desempeño, consistencia, cumplimiento, efectividad, etc.

MEDICIÓN DEL PROYECTO

Mediciones genéricas

La medición del proyecto y sus recursos asociados constituye el elemento principal sobre el que se basa el estudio de las métricas del proceso software. Cuando se mide el proyecto el objetivo fundamental que se pretende es el de reducir el coste total y el tiempo de desarrollo del mismo. Los indicadores de proyecto permiten al administrador de software (Pressman, 2001):

- Evaluar el estado del proyecto en curso.
- Realizar un seguimiento de los riesgos potenciales.

- Detectar las áreas de problemas antes de que se conviertan en “críticas”.
- Ajustar el flujo y las tareas de trabajo.
- Evaluar la habilidad del equipo del proyecto en controlar la calidad de los productos de trabajo de la ingeniería del software.

En relación a las métricas de proceso, las mediciones del proyecto de software se consideran a un nivel táctico, es decir, las métricas de proyectos y los indicadores derivados de ellos son utilizados por un administrador de proyectos y por un equipo de software para adaptar el flujo de trabajo del proyecto y las actividades técnicas.

Putnam y Myers (2003) establecen los siguientes aspectos a medir para la gestión de proyectos y cuyas métricas se denominan ***five core metrics***:

- Cantidad de Funcionalidad, obtenida a través de las métricas de tamaño (LOC, Puntos Función, etc.).
- Productividad, relación entre funcionalidad producida en el tiempo y el esfuerzo dedicado.
- Tiempo/Calendario. Duración del proyecto (usualmente en meses de calendario).
- Esfuerzo. Cantidad de trabajo en personas/mes.
- Fiabilidad. Expresada en ratio de defectos (o su métrica recíproca MTTD – Tiempo promedio entre defectos).

El primer tipo de métricas de proyectos software pueden ser obtenidas durante la fase de estimación. Las métricas recopiladas de proyectos anteriores se utilizan como la base a partir de la cual se realizan las estimaciones del esfuerzo y del tiempo necesario para el proyecto actual.

A medida que avanza un proyecto, las métricas del esfuerzo y del tiempo consumido se comparan con las estimaciones originales (y la planificación del proyecto). El administrador de proyectos utiliza estos datos para supervisar y controlar el avance. Para la estimación del tamaño del software cabe destacar la métrica de “*Puntos Función*” mientras que para la estimación de costes de un proyecto caben destacar los modelos COCOMO (COnstructive COst MOdel) (Boehm, 1981) y su posterior refinamiento en la versión actualmente en vigor COCOMO II, así como el modelo SLIM (Putnam y Myers, 2003).

En Desouza (2015) se recogen diferentes métricas para evaluar la gestión de proyectos (en el ámbito gubernamental) que se muestran en la Tabla 4.1.

Productividad del empleado

- Coste por contratación
- Beneficio por empleado
- Ratio de bajas
- Ratio coste/beneficio
- Satisfacción del empleado
- Ratio de rotación
- Costes de rotación
- Compensación a los trabajadores
- Tiempo para cubrir vacantes

Planificación

- Número de proyectos en progreso
- Número de proyectos en progreso a 30, 60, y 90 días
- Cumplimiento de la planificación
- Porcentaje de proyectos terminados a tiempo
- Porcentaje de proyectos terminados con retraso
- Tiempo medio de terminación de proyectos

Presupuesto

- Cantidad de presupuesto gastado en nuevos proyectos de TI

- Cantidad de presupuesto gastado en prototipado y experimentación con tecnologías emergentes
- Costes reales a la fecha
- Mayor línea de varianza ítem/cantidad
- Índice de desempeño de coste
- Índice de desempeño de planificación
- Porcentaje sobre presupuesto
- Presupuesto anual
- Valor planificado por proyecto
- Cambio en el presupuesto operativo respecto al año anterior
- Porcentaje de organizaciones que esperan gastar más de lo presupuestado en operaciones de TI
- Total gastado por usuario
- Total gastado en TI como porcentaje de ingresos
- Total de presupuesto del proyecto según lo planificado
- Cantidad por debajo del presupuesto
- Cantidad por encima del presupuesto

Satisfacción del cliente de la organización

- Métricas de proyecto adaptadas decididas colaborativamente por clientes, colaboradores y el departamento de TI
- Porcentaje de objetivos deseados conseguidos
- Satisfacción del cliente
- Valoraciones de la satisfacción del cliente por medio de encuestas
- Ratio de retención de clientes
- Número de personal de TI y no TI, para recoger la contribución a la organización global y el valor desde la perspectiva de TI
- Incidentes con SLA/total de incidentes
- Disponibilidad/fiabilidad de la infraestructura
- Soporte a operaciones
- Fidelidad del cliente

Tabla 4.1. Métricas para evaluar proyectos (Desouza, 2015)

Mediciones en proyectos ágiles

Los proyectos ágiles requieren métricas específicas de acuerdo a su naturaleza. En relación a las estimaciones sobre el esfuerzo necesario de desarrollo ágil, una de las métricas que se hizo más popular desde los inicios de la aplicación de métodos ágiles es la de “puntos historia” (Cohn, 2005).

Por su parte, en Kupiainen et al. (2015) se lleva a cabo una revisión de métricas utilizadas en proyectos ágiles, a partir de 774 artículos. Estos autores destacan que las métricas se utilizan por cinco importantes razones:

- Planificación de proyecto y *sprint*, es decir: priorización de las tareas (para lo que se utilizan la mayoría de las métricas), determinación del alcance (utilizándose las métricas para estimar el tamaño y el número de características que se desarrollarán, destacándose las métricas relacionadas con la velocidad), y determinación de recursos (que sirve para flexibilizar el desarrollo).
- Monitorización del progreso del proyecto y *sprint*, que se subdivide en cuatro categorías: métricas para monitorizar el progreso del proyecto, incrementar la visibilidad, controlar el logro de los objetivos, y balancear el flujo de trabajo.
- Entendimiento y mejora de la calidad, en la que se pueden encuadrar métricas usadas para entender el nivel de calidad antes y después de la liberación, métricas usadas para mejorar el nivel de calidad, y métricas usadas para asegurar el nivel de pruebas.

- Corrección de problemas de procesos software, para ayudar a entender y corregir problemas en los procesos de ingeniería del software.
- Motivación de las personas. Se han utilizado las métricas para motivar a las personas a reaccionar más rápidamente a los problemas.

En la Tabla 4.2 se resume para cada categoría las principales métricas que se utilizan. Remitimos al lector a Kupiainen et al. (2015) para localizar las fuentes en las que se explican todas estas métricas.

CATEGORÍA	MÉTRICAS
Planificación de proyecto y <i>sprint</i>	<ul style="list-style-type: none"> • Velocidad • Estimación de esfuerzo • Valor al cliente • Tiempo de entrega • Tareas realizadas/no realizadas • Fecha prevista de tarea realizada • N° previsto de defectos • Capacidades necesarias
Monitorización del progreso del proyecto y <i>sprint</i>	<ul style="list-style-type: none"> • Trabajo terminado • N° de pruebas automáticas pasadas • <i>Burndown</i> • <i>Check-ins</i> • Defectos • Tendencias de defectos • Porcentaje de historias terminadas • Tipos de coste • Ratio de requisitos por fase • Varianza en <i>handovers</i> • Cuadro de deuda técnica • Tiempo de ciclo

	<ul style="list-style-type: none"> • Tiempo común • Trabajo en progreso • Porcentaje de flujo de historias • Efectividad del equipo • Inventario de requisitos en el tiempo • Estimación del esfuerzo • Nº de requisitos por fase
Entendimiento y mejora de la calidad	<ul style="list-style-type: none"> • Nº de peticiones de cambio • Esfuerzo de mantenimiento • Puntuación de promotor neto • Defectos • Defectos diferidos • Defectos críticos enviados por los clientes • <i>Burndown</i> • <i>Check-ins</i> • Nº de pruebas automáticas pasadas • Estado del <i>build</i> y nº de pruebas unitarias • Cobertura de las pruebas • Ratio del crecimiento de las pruebas • Violación del análisis estático del código • Cuadro de deuda técnica • Trabajo en curso • Porcentaje de historias terminadas • Tiempo de ciclo
Corrección de problemas de procesos software	<ul style="list-style-type: none"> • Tiempo de entrega • Tiempo de procesamiento • Tiempo de cola • Tipos de coste • Ratio de requisitos por fase • Varianza en <i>handovers</i> • Nº de requisitos por fase

	<ul style="list-style-type: none"> • Porcentaje de flujo de historia • Tendencia de defectos • Índice de desempeño del coste (CPI) • Índice de desempeño del cronograma (SPI) • Porcentaje de historias terminadas • Trabajo en progreso • Inventario de requisitos en el tiempo • Velocidad • <i>Burndown</i> • Porcentaje de historias preparadas para el sprint • Nº de <i>bounce backs</i> • Nº de pruebas automáticas pasadas, • <i>Burndown</i> • <i>Check-ins</i> • Defectos • Nº de ítems de trabajo • Tiempo de corrección de <i>builds</i> fallidos, • Violación del análisis estático del código
Motivación de las personas	<ul style="list-style-type: none"> • Defectos • Tendencia de defectos • Tiempo de corrección de <i>builds</i> fallidos • Estado del <i>build</i> • Violación del análisis estático del código • Cuadro de deuda técnica • Nº de pruebas automáticas superadas • Trabajo en progreso • Velocidad

Tabla 4.2. Clasificación de las métricas ágiles según categorías.

En la Figura 4.2 se puede ver la distribución de métricas en función de la categoría.

Kupiainen et al. (2015) además destacan las métricas que tienen una mayor influencia en el desarrollo de software ágil, tal y como se muestra en la Tabla 4.3, se le asigna una ocurrencia e importancia a cada una de ellas. Para determinar la influencia de una métrica se utiliza, por un lado, el número de ocurrencias y por otro la importancia percibida en los estudios primarios. En general se evidencia que las métricas que tienen más influencia son las que se pueden calcular utilizando herramientas existentes, las que tienen capacidad de provocar debates o análisis de causas raíz, y las que proporcionan visibilidad de los problemas.

MÉTRICA	OCURRENCIA	IMPORTANCIA
Velocidad	15	3
Estimación de esfuerzo	12	3
Satisfacción del cliente	6	3
Total de defectos	8	2
Deuda técnica	2	3
Estado del <i>build</i>	2	3
Progreso como código operativo	1	3
Tiempo de entrega	4	2
Porcentaje de flujo de historia	1	2
Velocidad de elaborar características	1	2
Porcentaje de historias terminadas	1	2
Nº de casos de prueba	1	2
Tiempo de cola	1	2

Tiempo de procesamiento	1	2
Indicador de tendencia de defectos	1	2
Trabajo en progreso	6	1
Nº de pruebas unitarias	5	1
Tipos de coste	1	1
Varianza en handovers	1	1
Defectos diferidos	1	1
Nº previsto de defectos en el backlog	1	1
Cobertura de pruebas	1	1
Ratio de crecimiento de pruebas	1	1
Check-ins por día	3	NA
Tiempo de ciclo	2	NA

Tabla 4.3. Ocurrencia e importancia de las métricas ágiles.

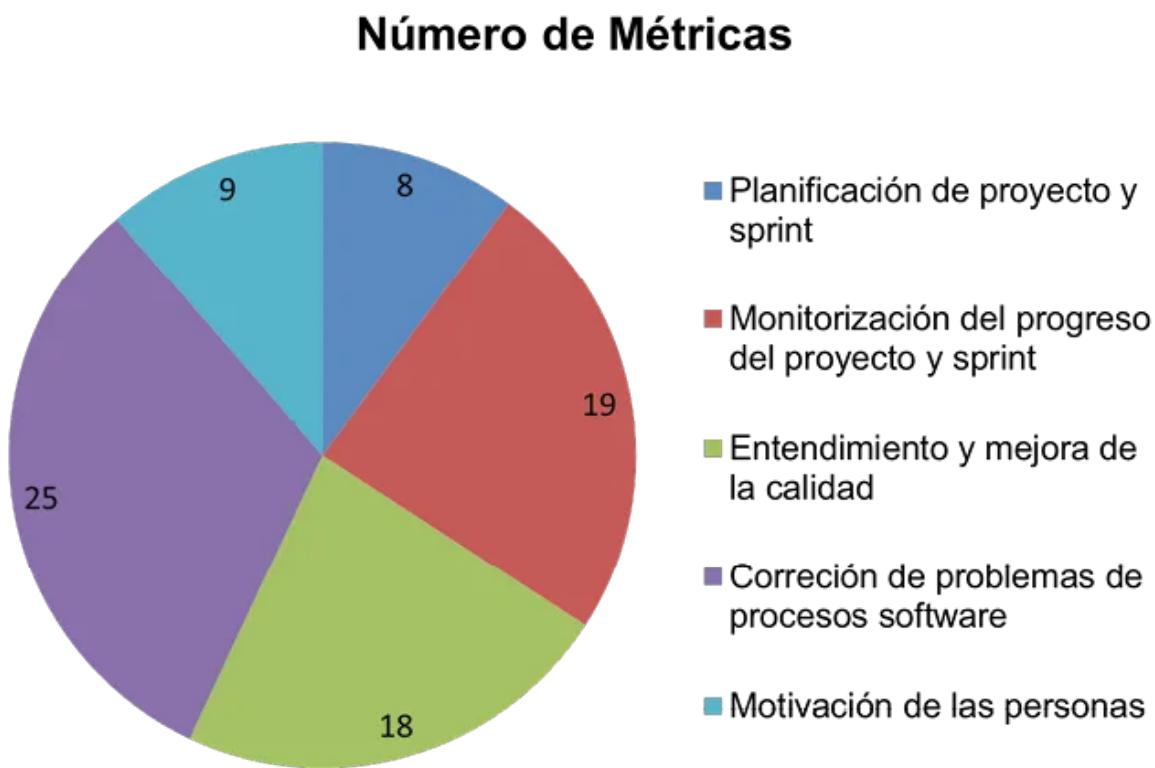


Figura 4.2 Distribución de las métricas por categorías

Mediciones en proyectos DevOps

Los proyectos DevOps tienen una naturaleza iterativa en el que se sigue un enfoque de mejora continua basada en la realimentación recibida; son muy colaborativos; afecta a toda la organización, sus procesos y tecnología; y están altamente automatizados donde se busca entrega rápida, continua, con servicios seguros y efectivos en coste (Gartner, 2015). Sobre todo se requiere de una gran sinergia y colaboración entre los departamentos de desarrollo y operaciones, la capacidad de producir nuevas versiones software en periodos menores de tiempo, y automatizar en lo posible el soporte a lo anterior, como por ejemplo proporcionando sistemas de tipo “infrastructure as code”.

Davis (2015) destaca que hay que mantener la filosofía “continua” no sólo en la integración continua o la entrega continua, sino también en las métricas, para lo que propone un ciclo continuo como se muestra en la Figura 4.3.

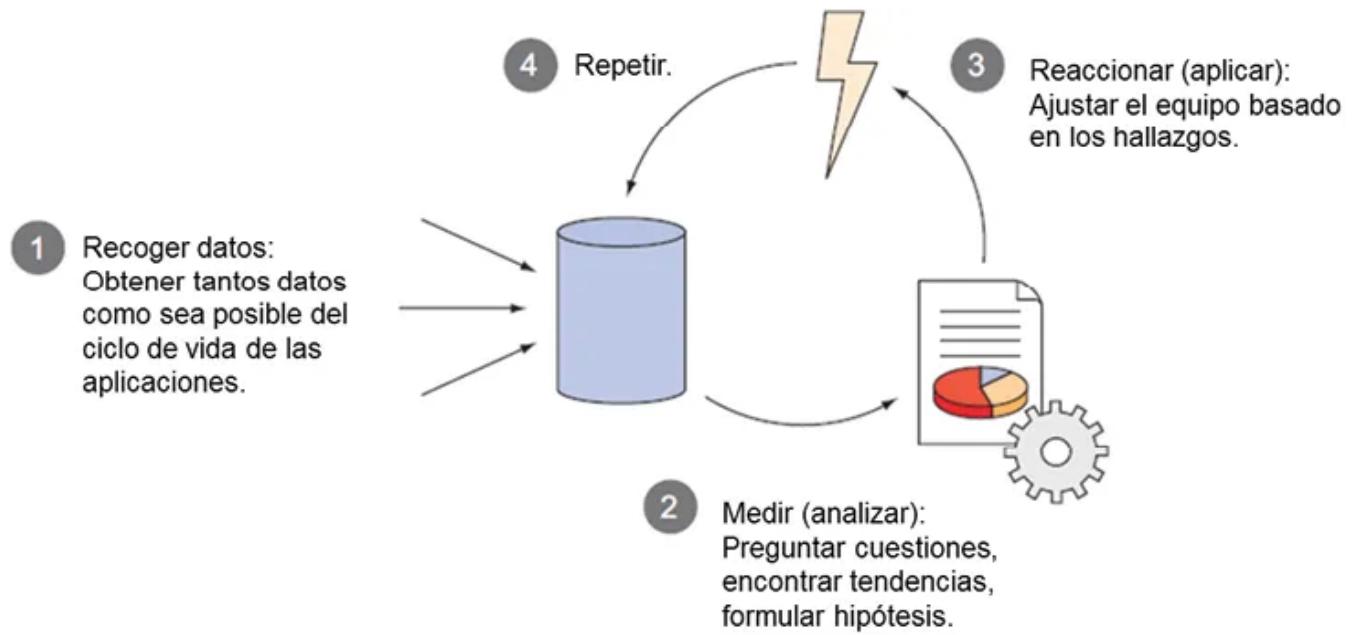


Figura 4.3 Ciclo continuo basado en métricas (Davis, 2015).

También propone varias métricas basándose en los diferentes componentes del ciclo DevOps (como los que se muestran en la Figura

4.4).

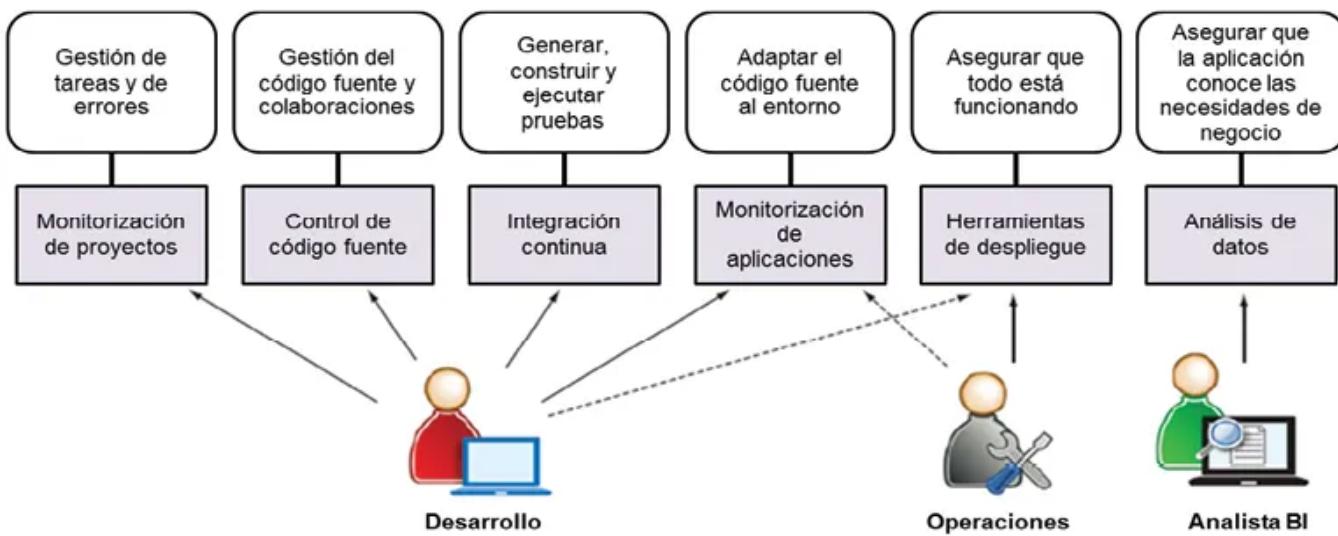


Figura 4.4 Componentes del ciclo DevOps (Davis, 2015).

- **Métricas de monitorización de proyectos**

Las métricas típicas utilizando datos del sistema de monitorización de proyectos son:

- *Burndown* (o diagrama de quemado), es una representación gráfica del trabajo por hacer en un proyecto en el tiempo.
- Velocidad, calculada teniendo en cuenta el total que se estima con el total realizado.
- Flujo acumulativo, que muestra cuánto trabajo agregado por tipo se asigna al equipo en el tiempo.
- Tiempo de entrega (*lead time*), como total de tiempo entre que una tarea empieza y termina.
- Nº de errores, que representan inconsistencias en el software.

Las métricas clave de gestión de proyectos que habrá que combinar para obtener una visión del desempeño del equipo son:

- Estimaciones, el total de esfuerzo percibido que un equipo asigna a una tarea antes de trabajar en ella.
 - Volumen, número de tareas que se terminan.
 - Errores, número de defectos que se crean y se trabajan por parte de un equipo.
 - Reincidencia, tareas que alguien dijo que eran lo suficientemente buenas para hacer progresar en el proceso pero que terminaron por hacerlo retroceder.
- **Métricas de control de código fuente**

Si se considera que en la actualidad en el desarrollo de software se suele trabajar en entornos DVCS (*Distributed Version Control Systems*), una práctica habitual es la de las peticiones de “pull”. Es decir, cuando un desarrollador tiene un cambio o un conjunto de cambios que quiere combinar con la base de código maestra, envía sus cambios mediante una petición de “pull”. Esta petición, incluye una lista de otros desarrolladores que deberían revisar la petición, antes de aprobarla o denegarla.

Por tanto, de la gestión de código fuente podemos extraer las siguientes métricas clave:

- Peticiones de pull
- Peticiones de pull denegadas
- Peticiones de pull combinadas
- Commits
- Revisiones

Todas estas métricas se pueden combinar en un gráfico como el que se muestra en la Figura 4.5.

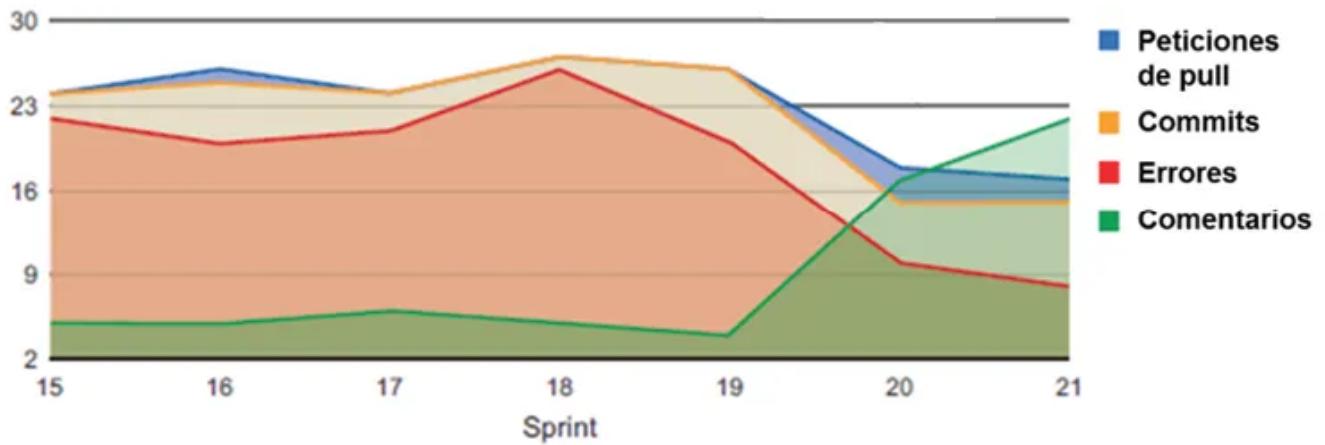


Figura 4.5 Métricas de control de código fuente (Davis, 2015).

- **Métricas de integración continua (y despliegue)**

La información básica que se puede obtener de la integración continua es la de construcciones (*builds*) exitosas y fallidas (véase Figura 4.6). Además, podemos encontrarnos con métricas procedentes de diferentes herramientas como: herramientas de gestión de pruebas de diferentes tipos (unitarias, de integración, de estrés, etc.) y de análisis estático (ej. SonarQube). Con todo esto podemos obtener información de:

- Informes de pruebas
- Número total de pruebas
- Porcentaje de pruebas exitosas y fallidas
- Análisis estático de código
- Porcentaje de cobertura de las pruebas
- Violaciones de código

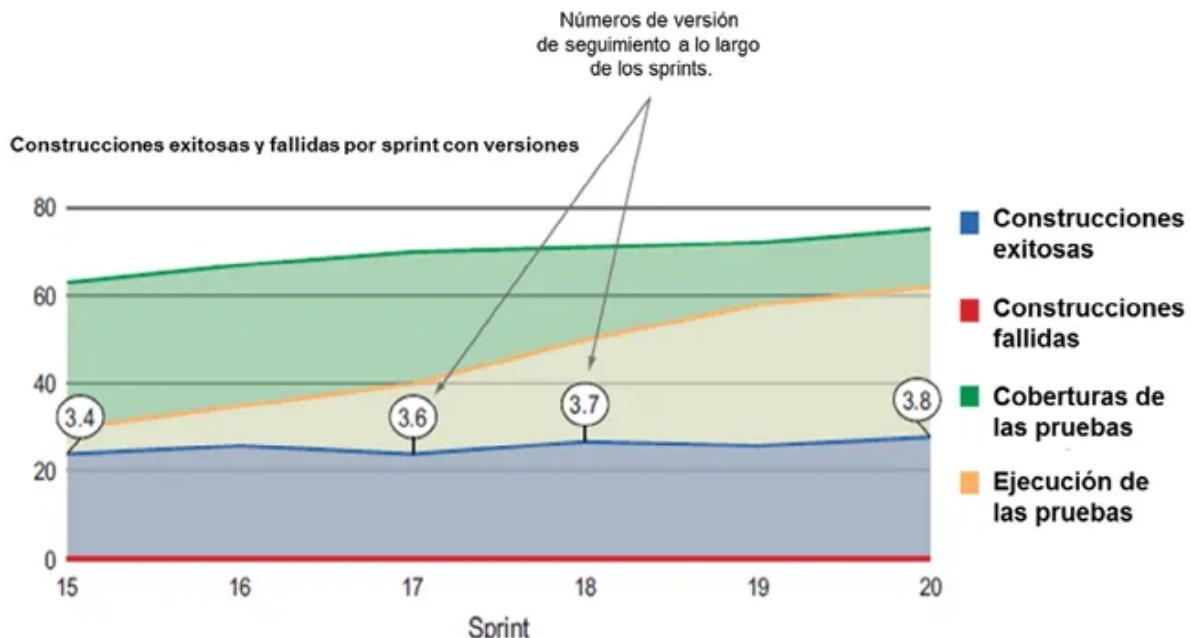


Figura 4.6 Métricas de integración continua (Davis, 2015).

- **Métricas de monitorización de aplicaciones (y análisis de datos)**

Los sistemas de monitorización permiten recabar métricas acerca de:

- Conexiones de red
- Utilización de CPU
- Utilización de memoria
- Transacciones
- Conexiones a bases de datos
- Espacio de disco
- Recolección de basura (“garbage”)
- Cuenta de los hilos (*threads*)
- Tamaño de la pila (*heap*)
- Ratios de error
- Tiempos de respuesta

Nicolette (2015) distingue entre indicadores “trailing” o “lagging” que son los que proporcionan información sobre lo que ya ha pasado, e indicadores de “leading” que son los que ayudan a predecir cosas. En particular propone las siguientes métricas para enfoques adaptativos, como es el caso de los entornos DevOps, a las que añade sus correspondientes representaciones gráficas.

1. Porcentaje de alcance terminado (*percentage of scope complete*)

Esta métrica sirve para saber si el proyecto se encuentra de acuerdo a la planificación, es decir, refleja la cantidad de trabajo planificado que ha sido terminada en la fecha, y puede ser un indicador de riesgos potenciales de entrega. En el caso de los entornos más adaptativos como DevOps, aunque siguen una gestión más ligera, también tienen una lista de características en forma de product backlog, lista de historias de usuario, o cola de trabajos, que se usan como base para el cálculo de esta métrica. Su representación gráfica se muestra en la Figura 4.7.



Figura 4.7 Alcance terminado hasta la fecha (Davis, 2015)

2. Agotamiento de presupuesto (*budget burn*)

Esta métrica sirve para conocer si se dispone de los suficientes recursos como para terminar el trabajo planificado según lo previsto. Refleja el rendimiento del presupuesto que se puede predecir basándose en el presupuesto real gastado hasta la fecha, por lo que advierte sobre posibles sobrecostes.

Su representación gráfica se muestra en la Figura 4.8.

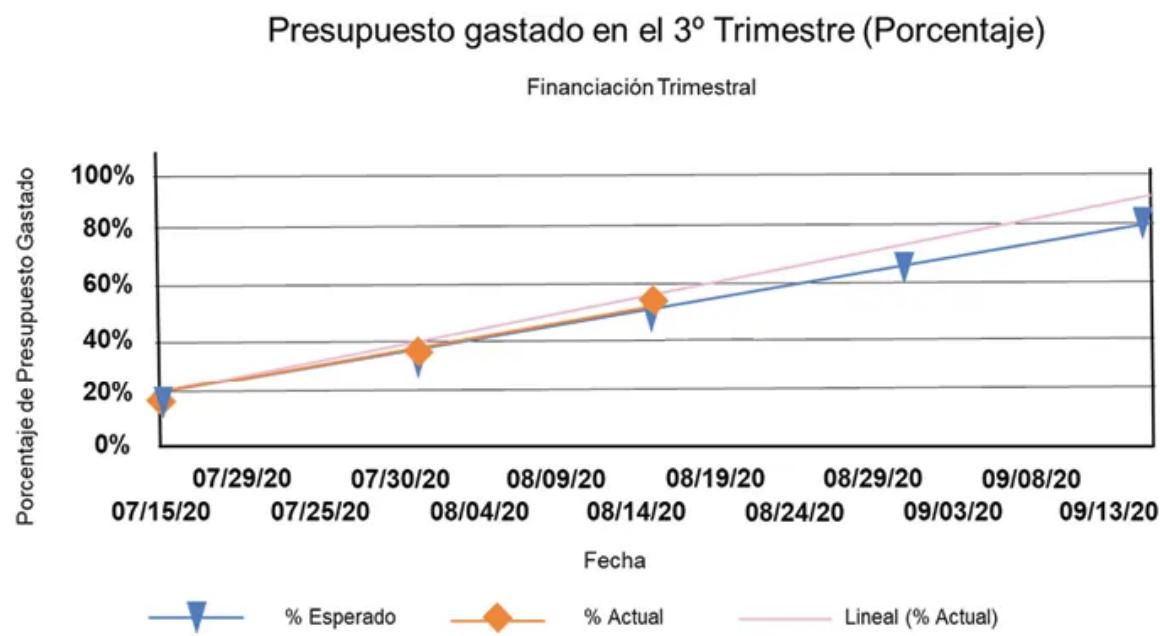


Figura 4.8 Agotamiento de presupuesto (Davis, 2015)

3. Ratio de agotamiento del buffer (*buffer burn rate*)

Esta métrica permite conocer si sobrepasaremos el buffer de planificación (es decir el margen que prevemos en una planificación, que suele estar -según este autor- en un 35%), lo cual permite monitorizar el ratio de agotamiento del buffer, y ser conscientes de tendencias que indican la presencia de riesgos de entrega (zonas amarillas y rojas). Su representación gráfica se muestra en la Figura 4.9.

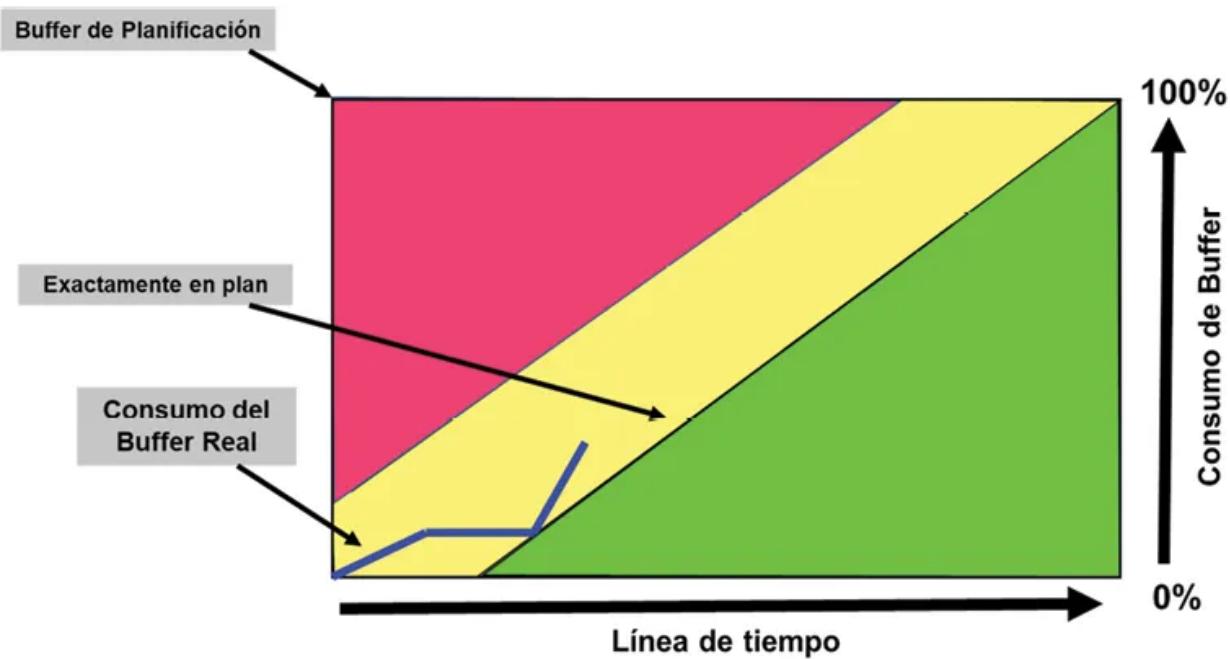


Figura 4.9 Ratio de agotamiento del buffer (Nicolette, 2015)

4. Características probadas en ejecución (*running tested features*)

Esta métrica permite conocer cuántas de las características planificadas de la solución se encuentran en un estado listo de producción, por lo que permite conocer si se crean regresiones (fallos en software que funcionaba bien) cuando se despliegan nuevas características, y pueden servir también para estimar si se completará funcionalidad suficiente a tiempo como para justificar la continuación del proyecto, o el tiempo necesario para terminar un conjunto determinado de características para una nueva solución. Su representación gráfica se muestra en la Figura 4.10.

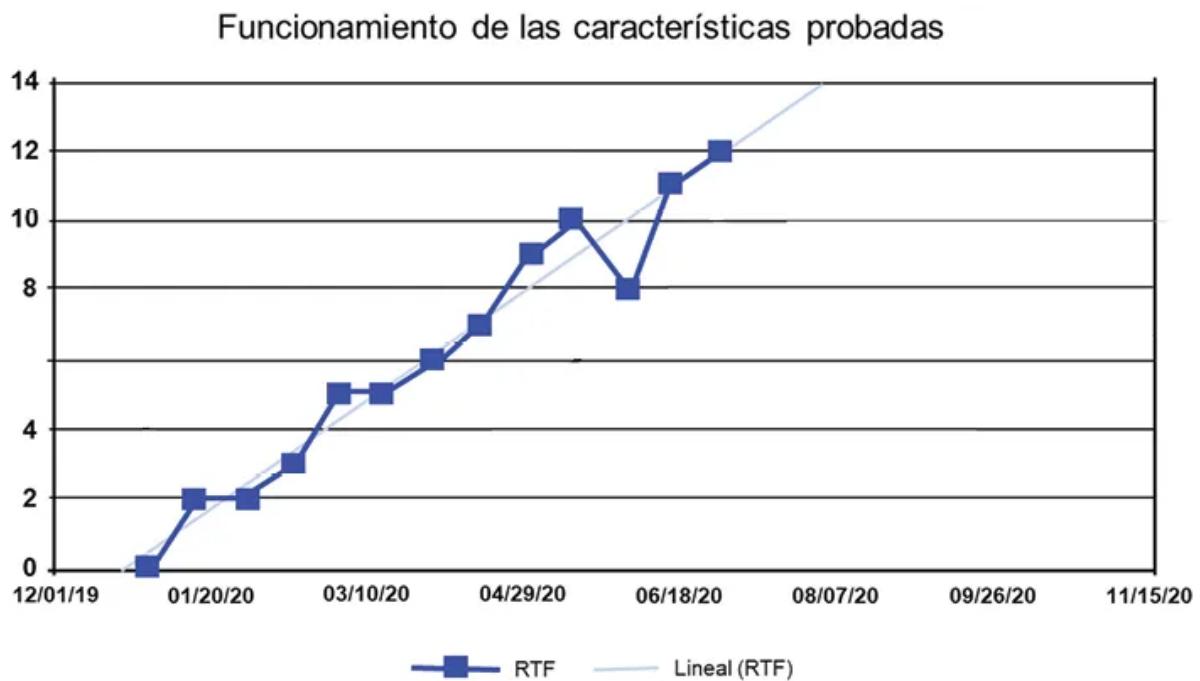


Figura 4.10 Características probadas en ejecución (Nicolette, 2015)

5. Valor de negocio ganado (*earned business value*)

Esta métrica refleja la proporción del valor que se ha entregado hasta la fecha, por lo que permite conocer si se han conseguido los objetivos del proyecto de manera que se pueda afirmar que estamos cumpliendo satisfactoriamente y podemos seguir adelante. También sirve, por tanto, para saber si vale la pena el coste de continuar desarrollando las características que quedan. Su representación gráfica se muestra en la Figura 4.11.

Valor de Negocio Ganado (*Earned Business Value*)

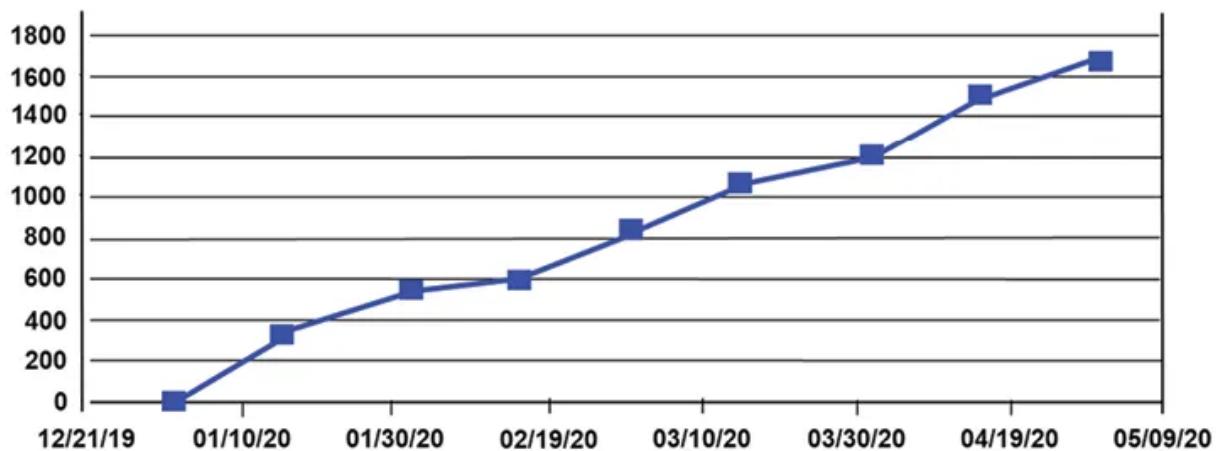


Figura 4.11 Valor de Negocio Ganado (Nicolette, 2015)

6. Velocidad (*velocity*)

Esta métrica representa la capacidad de entrega media del equipo por unidad de tiempo, es decir, podría utilizarse para saber si el equipo está entregando software a un ratio adecuado. Por tanto, esta métrica se puede utilizar para predecir el tiempo que el equipo necesitará para terminar de entregar una determinada funcionalidad. Su representación gráfica se muestra en la Figura 4.12.

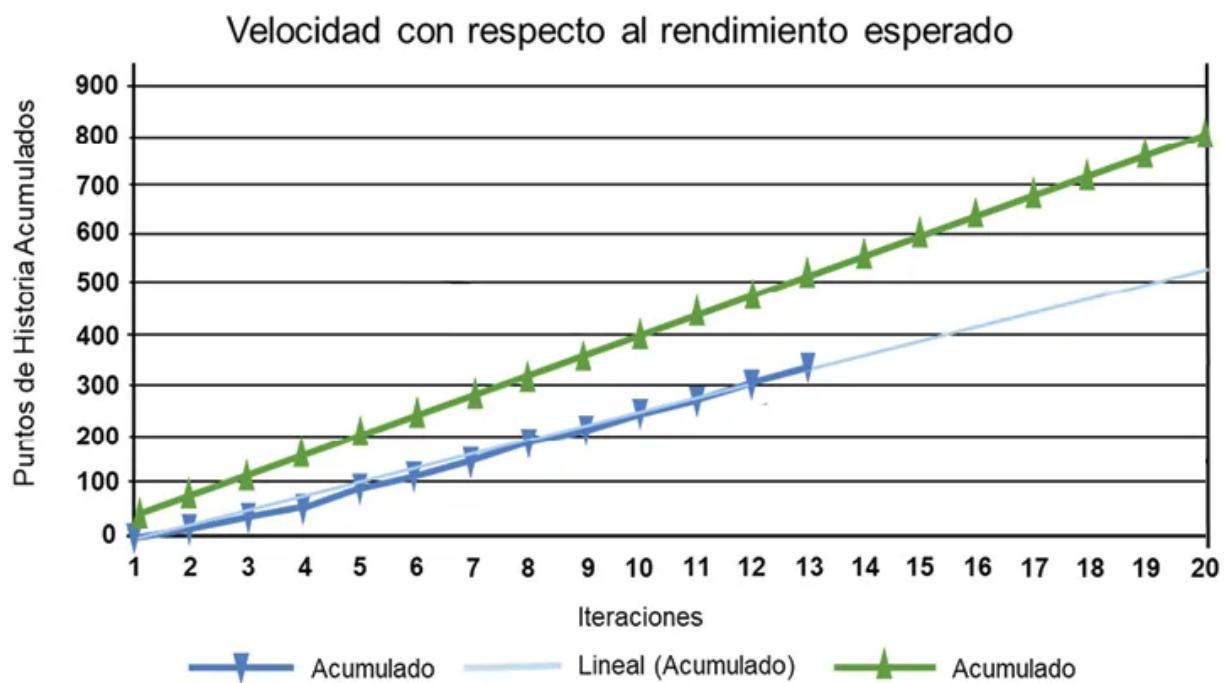
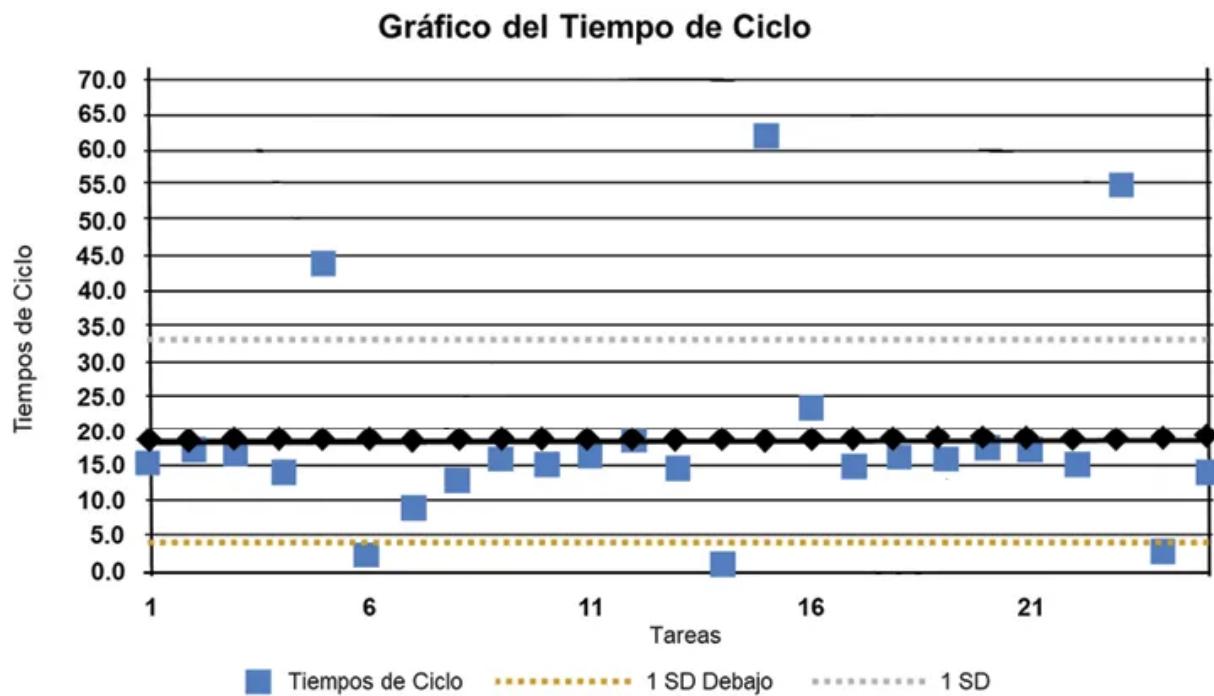


Figura 4.12 Velocidad (Nicolette, 2015)

7. Tiempo de ciclo (*cycle time*)

Esta métrica permite estimar el tiempo medio necesario para terminar un determinado elemento de trabajo, por lo que permite determinar la consistencia del desempeño de entrega del equipo. Su representación gráfica se muestra en la Figura 4.13, donde se analiza la desviación estándar (SD) de los valores para determinar si son variaciones debida a causas comunes (normal) o debida a causas especiales (outliers mostrados en la parte superior de la figura). Véase el capítulo 5 para una descripción más detallada de los gráficos de control.



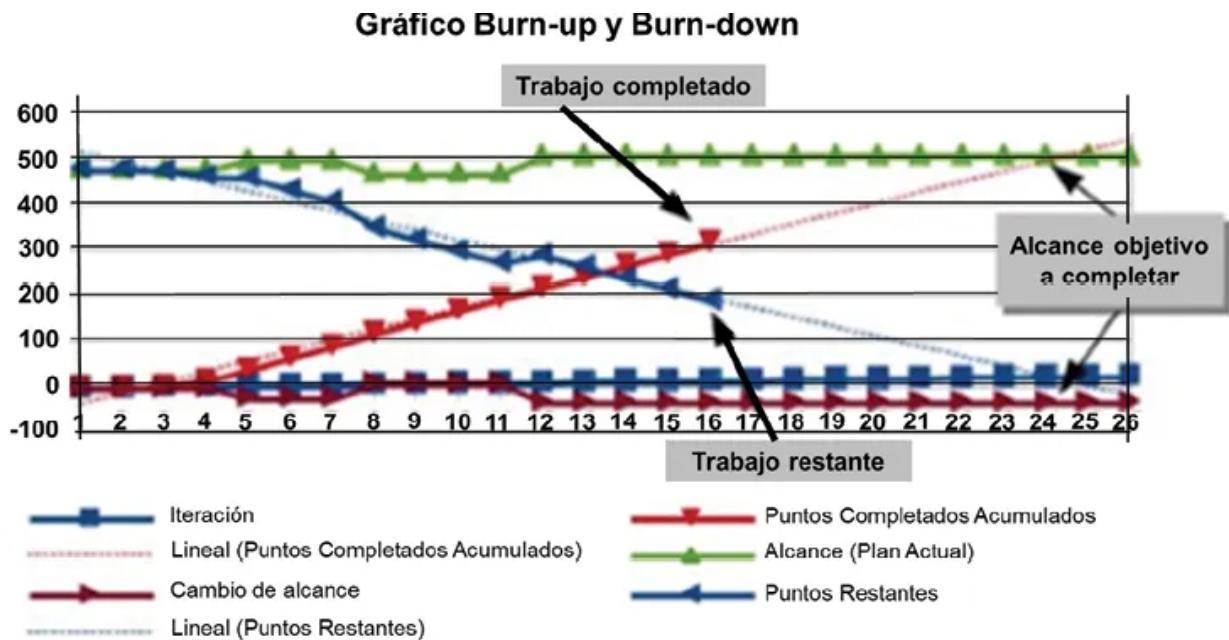


Figura 4.14 Diagrama de Burn-Up y Burn-Down (Nicolette, 2015)

9. Productividad (*throughput*)

Esta métrica determina cuánto software puede entregar un equipo en un determinado tiempo, y por tanto indica si el equipo entrega resultados a un ratio consistente. Su representación gráfica se muestra en la Figura 4.15.

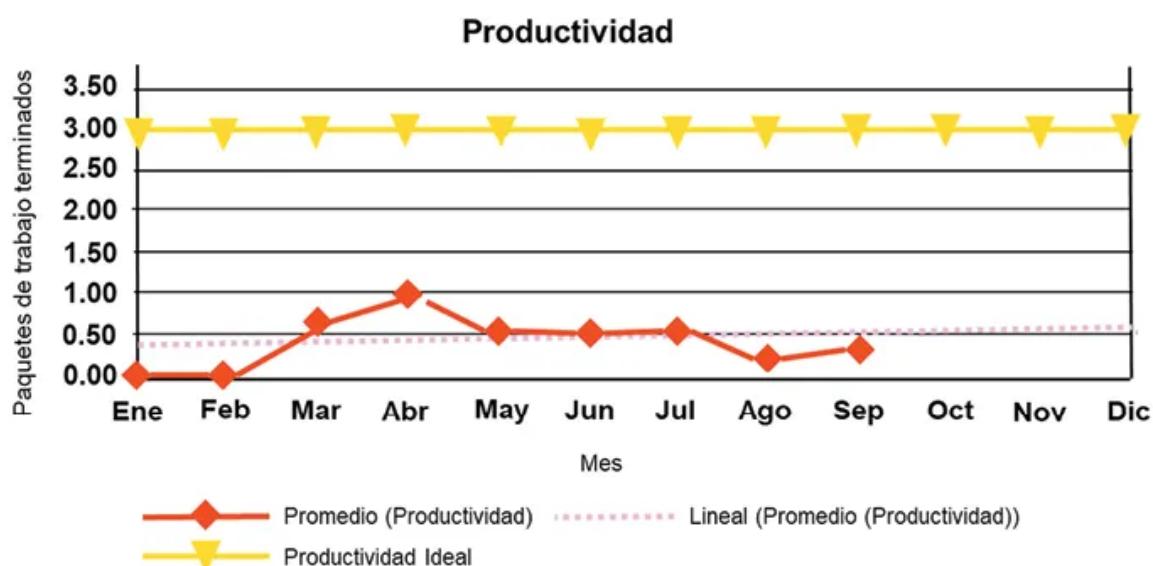


Figura 4.15 Productividad (Nicolette, 2015)

10. Flujo acumulado (*cumulative flow*)

Esta métrica permite conocer los cuellos de botella del proceso, el tipo de colas que se forman en el proceso, e incluso los puntos en los que la carga de trabajo no está balanceada. Su representación gráfica se muestra en la Figura 4.16.

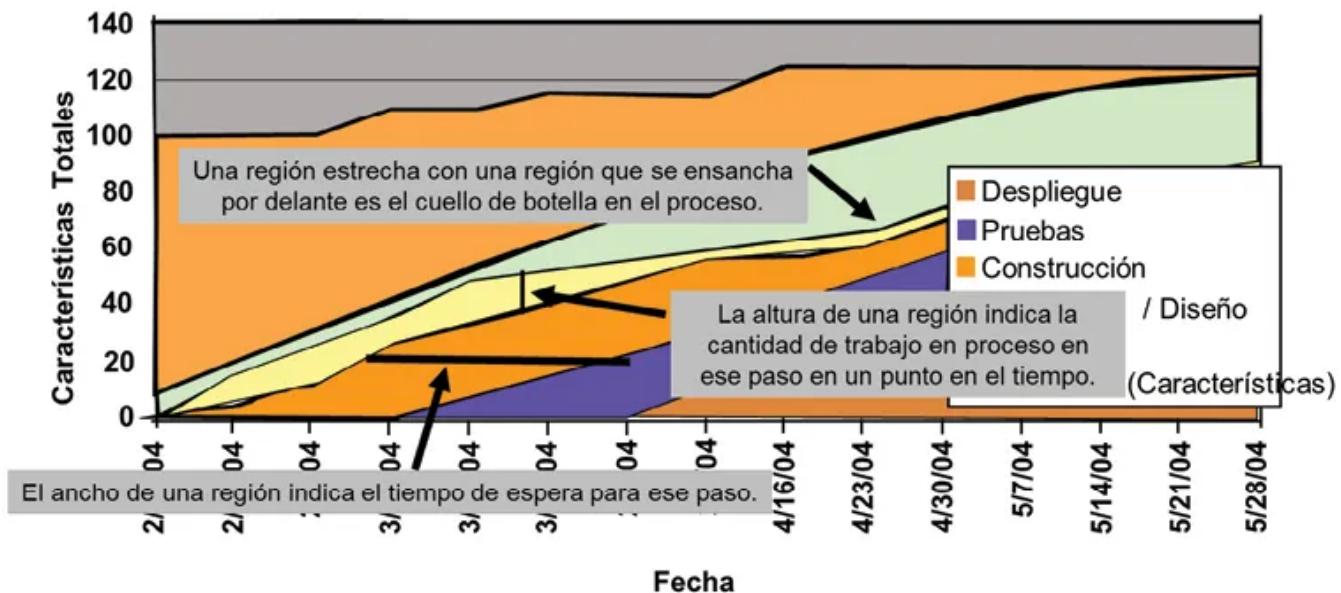


Figura 4.16 Flujo acumulado (Nicolette, 2015)

MEDICIÓN DEL PRODUCTO

La medición del producto software está centrada en evaluar la calidad de los entregables. Los productos del software son las salidas del proceso de producción del software, que incluyen todos los artefactos entregados o documentos que son productos durante el ciclo de vida del software. En la literatura existe una gran diversidad de propuestas relacionadas con la medición del producto, la mayor parte a nivel de código o de diseño de bajo nivel.

Métricas “Clásicas”

Entre las métricas tradicionales de producto cabe destacar las métricas de código fuente, siendo las más representativas la de Líneas de Código y de Longitud Total.

Líneas de código (LOC, *Lines of Code*) es la métrica más popular a nivel de código de programa. Sin embargo, a pesar de ser ampliamente conocida y utilizada, el problema de esta métrica ha sido la falta de consenso existente a la hora de definir qué es una línea de código, ya que esta definición variará en función de las necesidades o de la persona que la aplique. Por ejemplo, según el objetivo perseguido por la medición será importante contar las líneas de comentario como líneas de código mientras que en otras ocasiones será imprescindible no contar los comentarios como líneas de código. Por ello, para aplicar esta métrica es fundamental establecer claramente qué elementos hay que considerar como línea de código y cómo deben contarse. En particular es necesario clarificar elementos como las líneas en blanco, los comentarios, las declaraciones de datos y las líneas que contienen instrucciones separadas. En general se recomienda separar en la medición de la longitud las líneas de código y los comentarios.

Se define Longitud Total (LT) como la suma del Número de Líneas de Código que no son comentarios (NCLOC) más el número de líneas de código que son comentarios (CLOC).

A partir de la métrica anterior se pueden definir otras métricas derivadas útiles, como la densidad de comentarios (CLOC/LOC), que puede dar una idea sobre el punto hasta el cual está documentado el código.

Para facilitar la obtención e interpretación de la métrica LOC, el SEI ha definido listas de comprobación (Park, 1992) en las que se indica que

como línea de código se debe considerar todo el código ejecutable, declaraciones no ejecutables y directivas de compilación, pero no las líneas en blanco. También se debe considerar en la medición la forma en la que el código ha sido producido (programando, usando generadores de código fuente, copiado o reutilizado sin realizar cambios, modificado o convertido con traductores automáticos).

Otras métricas definidas para evaluar la longitud de un programa son:

- **Número de sentencias de programación.** Presenta el mismo tipo de problemas de ambigüedad que la métrica LOC.
- **SIZE1. Definida como el número de puntos y coma** (Li y Henry, 1993). Se creó intentando paliar el problema de ambigüedad de definición de las líneas de código. Como se puede deducir, esta métrica sólo es aplicable a programas que utilicen este símbolo para separar unas sentencias de otras.
- **Métricas de la Ciencia del Software (Software Science).** Propuestas por (Halstead, 1977) para intentar independizar las métricas del lenguaje de programación. Se basan en los *tokens* (unidades sintácticas elementales distinguibles por el compilador) y que pueden ser divididos en operadores y operandos. Las métricas son la longitud de un programa, el volumen de un programa y el esfuerzo de implementación de un programa. Las métricas base definidas para estos elementos son:
 - μ_1 : el número de operadores diferentes que aparecen en el programa
 - μ_2 : el número de operandos diferentes que aparecen en el programa
 - N_1 : el número total de veces que aparece el operador

- N2: el número total de veces que aparece el operando

A partir de las métricas base, Halstead define una serie de métricas derivadas que permiten evaluar: la longitud global del programa; el volumen mínimo potencial para un algoritmo; el volumen real (número de bits requeridos para especificar un programa); el nivel del programa (una medida de la complejidad del software); nivel del lenguaje (una constante para un lenguaje dado); y otras características tales como esfuerzo de desarrollo, tiempo de desarrollo e incluso el número esperado de fallos en el software. Como métrica más representativa para evaluar la longitud, Halstead propone la métrica N, que es la suma de N1 y N2. Del resto de métricas cabe destacar:

- Volumen de un programa: $V = N \cdot \log_2 (\mu)$, siendo $\mu = \mu_1 + \mu_2$.
- Longitud global del programa: $N' = \mu_1 \cdot \log_2 (\mu_1) + \mu_2 \cdot \log_2 (\mu_2)$

Fenton y Pfleeger (1997) consideran que, aunque la propuesta de Halstead ha tenido un gran impacto en la medición software, constituye un ejemplo de medición inadecuada, al proporcionar algunas métricas con definiciones confusas lo que puede provocar diversas interpretaciones de las mismas.

Entre las métricas clásicas de complejidad destacan las siguientes:

- **Complejidad Ciclomática (V(G))**, propuesta por McCabe (1976), para evaluar la complejidad de un programa. Esta métrica es, además de la primera métrica conocida, una de las más estudiadas y utilizadas. La métrica V(G) está basada en la teoría de grafos y mide el número de caminos linealmente independientes de un programa, que puede representarse mediante un grafo de flujo de control. Las estructuras típicas de un programa se

representan mediante un grafo de flujo de control de acuerdo al esquema mostrado en la Figura 4.17:

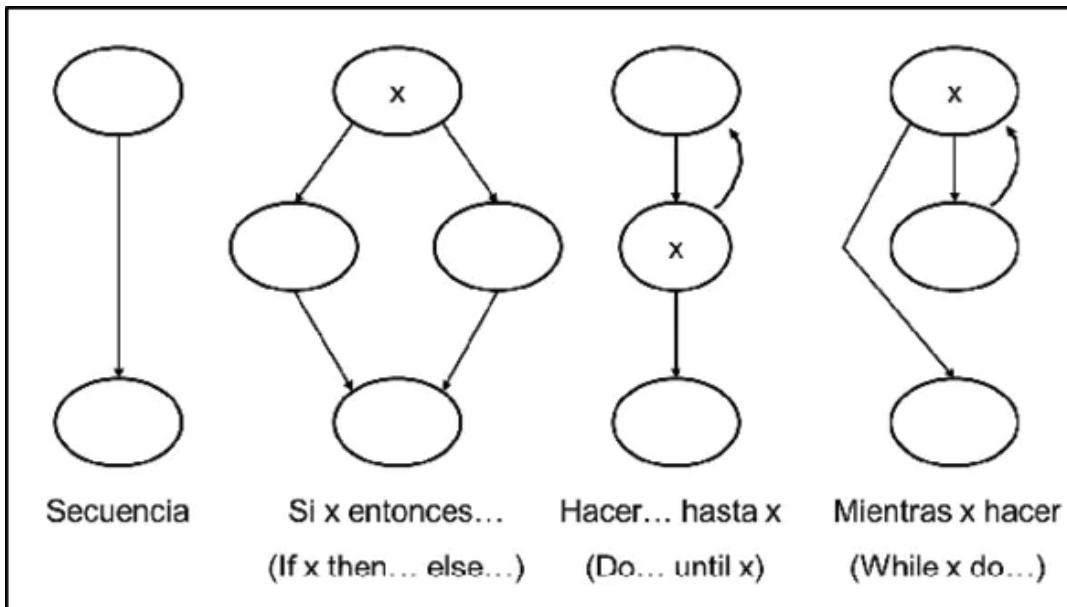


Figura 4.17 Representación de las estructuras básicas de un programa

A partir de un grafo de control, las formas alternativas de cálculo de esta métrica son las siguientes:

1. $V(G) = A - N + 2$, siendo A el número de arcos del grafo y N el número de nodos.
2. $V(G) = r$, siendo r el número de regiones cerradas del grafo.
3. $V(G) = c + 1$, siendo c el número de nodos de condición.

A mayor valor de la métrica $V(G)$ mayor complejidad del programa. McCabe indicó también que un valor razonable de esta métrica para que un módulo fuera mantenible debía ser menor de diez. Diversos estudios empíricos han encontrado una fuerte correlación entre la métrica de McCabe y el número de errores que existen en el código fuente, así como el tiempo requerido para encontrar y corregir dichos errores. Otra de las aplicaciones de esta métrica ha sido a nivel de

pruebas del software ya que puede utilizarse para predecir la cantidad de esfuerzo necesario para probar un módulo o programa.

Como ejemplo de cálculo de esta métrica considérese el ejemplo del programa en pseudocódigo mostrado en la Figura 4.18 (Piattini et al., 2003), junto con su representación de su grafo de control.

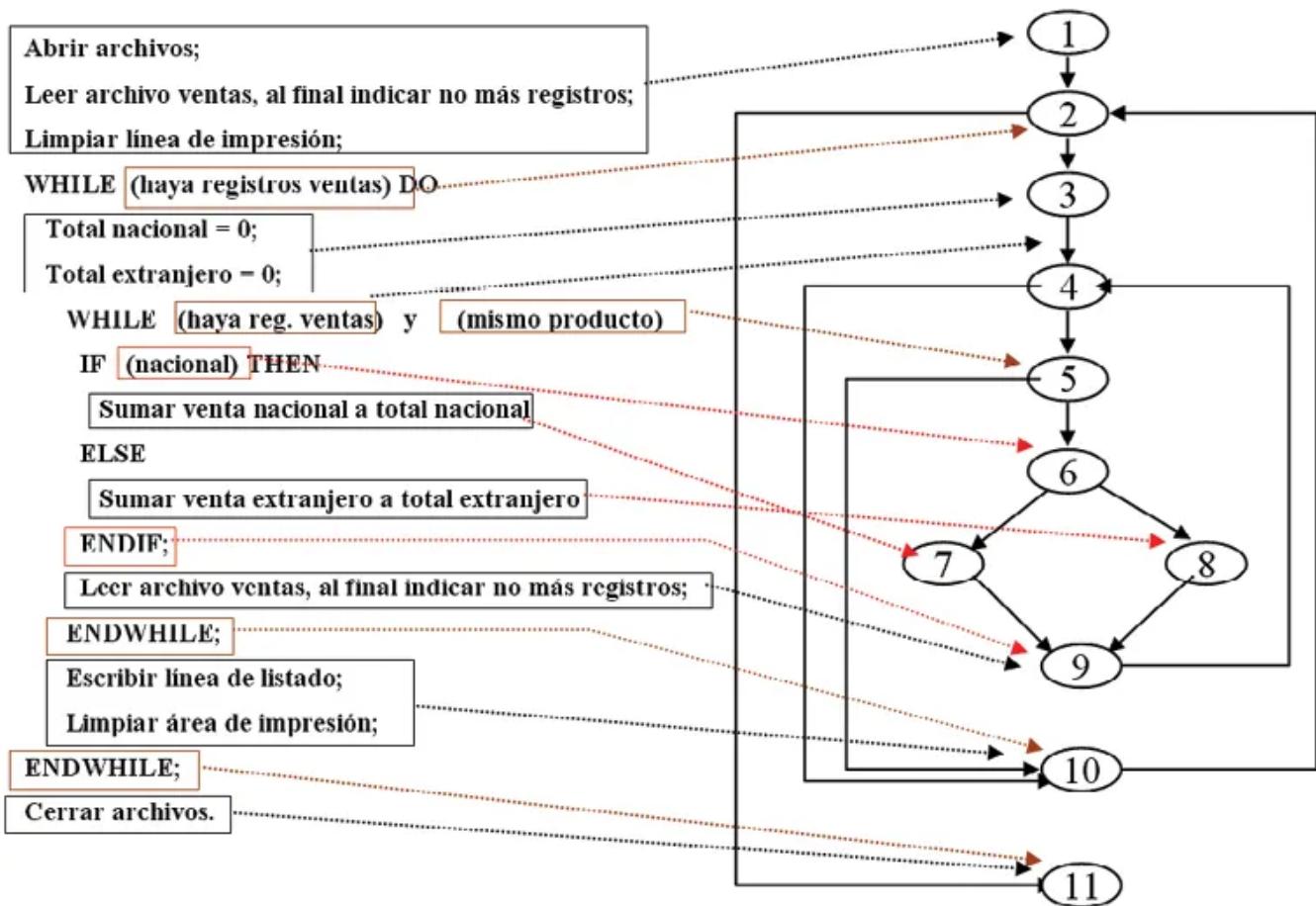


Figura 4.18 Ejemplo de un grafo de control

El valor de la métrica de McCabe para el ejemplo ilustrado en la Figura 4.18 sería el siguiente:

1. $V(G) = 14 - 11 + 2(A - N + 2) = 5$
2. $V(G) = 5$ regiones cerradas
3. $V(G) = 4 + 1 = 5(c + 1)$

- **Fan-in (concentración) y fan-out (expansión).** Propuestas por Henry y Kafura (1981) trabajan sobre la estructura de un módulo representada como un árbol o grafo de llamadas entre módulos. El *fan-in* de un módulo m es el número de flujos que terminan en m mientras que el *fan-out* es el número de flujos que salen de m .
- **Complejidad de un módulo**, que está basada en las dos métricas anteriores y creada también por Henry y Kafura (1981) siendo su definición: $MHK = longitud(i) \cdot [fan-in(i) * fan-out(i)]^2$, donde la longitud (i) es el número de sentencias en lenguaje de programación en el módulo i .

Henry y Kafura amplían la definición de concentración y expansión no sólo como el número de conexiones de control del módulo (llamadas al módulo), sino también el número de estructuras de datos del que el módulo i reúne (concentración) o actualiza (expansión) datos.

En la Figura 4.19 se muestra un ejemplo del diagrama de estructura que representa el diseño de un programa de gestión de préstamos bibliotecarios. El cálculo de las métricas para dicho ejemplo se muestra en la Tabla 4.4, en el que se indica la longitud de los distintos módulos.

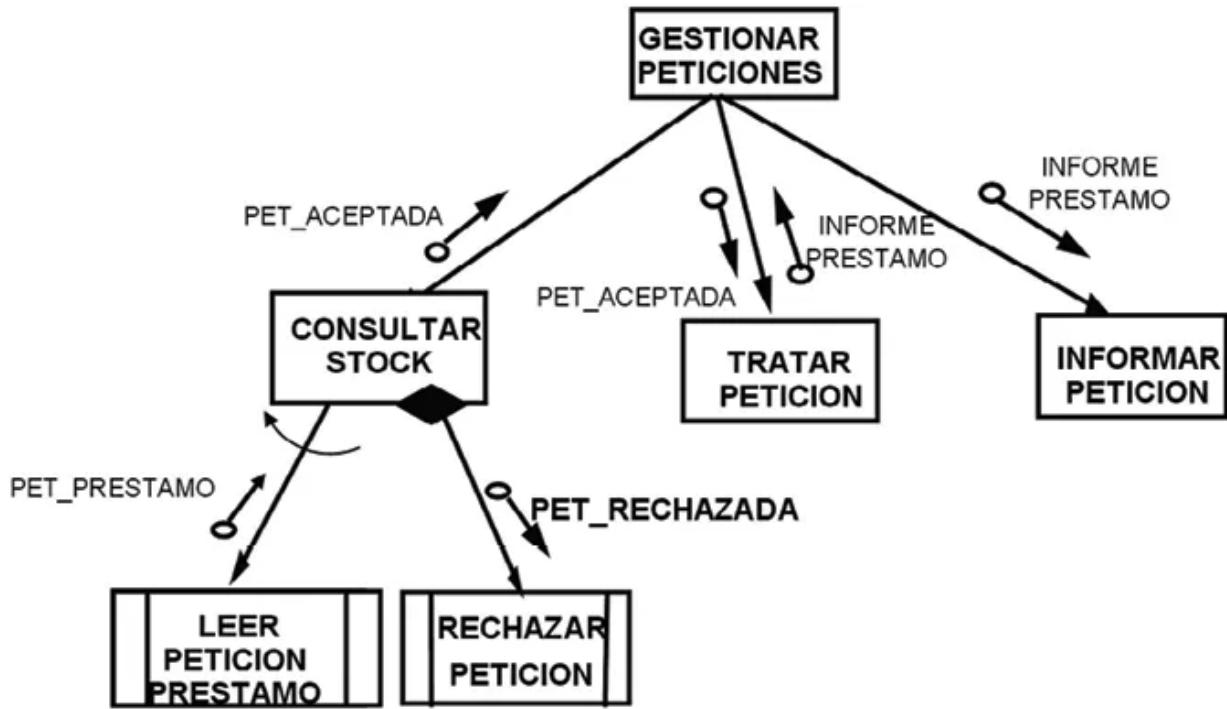


Figura 4.19 Programa de gestión de préstamos bibliotecarios. Diagrama de estructura

Módulo	Fan-In	Fan-Out	[Fin*FOut] ²	Longitud	MHK
Gestionar Peticiones	2	2	16	10	160
Consultar Stock	1	1	1	14	14
Tratar Petición	1	1	1	20	20
Informar Petición	1	0	0	6	0
Leer Petición Préstamo	0	1	0	8	0
Rechazar Petición	1	0	0	5	0

Tabla 4.4. Valores de las métricas de Henry y Kafura (1981)

Métricas para sistemas orientados a objetos

Aunque las métricas anteriores también se pueden usar para medir sistemas orientados a objetos (por ejemplo, el número de líneas de código de un programa escrito en Java), el software desarrollado siguiendo el paradigma OO difiere del desarrollado utilizando enfoques

tradicionales. Ello planteó la necesidad de definir nuevas métricas adaptadas a las características particulares de este paradigma, de las cuales las métricas MOOSE son las más importantes. Una revisión reciente (Canedo et al., 2019) que las métricas MOOSE (junto con las LOC y la Complejidad ciclomática) son las métricas más presentes en las herramientas de medición.

Otras métricas OO han sido las propuestas por Carbone y Santucci (2002), Brito e Abreu y Carapuça (1994) o Lorenz y Kidd (1994).

Métricas MOOSE

Las métricas MOOSE o también conocidas como métricas CK (Chidamber y Kemerer, 1994) son las más difundidas en orientación a objetos. De hecho, existen numerosos trabajos empíricos sobre la relación de estas métricas y, por ejemplo, la propensión a errores, o la mantenibilidad de las clases, etc. Este conjunto de métricas, definidas a nivel de clases, está compuesto por las siguientes seis métricas:

- **Métodos ponderados por clase** (WMC, *Weighted Methods per Class*), que mide la complejidad de una clase. Si todos los métodos son igualmente complejos, entonces WMC es igual al número de métodos definidos en una clase. Sea la clase C_i que tiene los métodos M_1, \dots, M_n siendo su complejidad respectiva c_1, \dots, c_n , es posible definir la fórmula:

$$WMC = \sum_{i=1}^n C_i$$

Si consideramos en el ejemplo de la Figura 4.20 que todos los métodos tienen complejidad 1, entonces: WMC(Persona) = 8, WMC(Cliente)= 4

(no considerando los métodos heredados) y $WMC(\text{Empleado})=10$ (no considerando los métodos heredados).

- **Profundidad del Árbol de Herencia de una Clase, (DIT, *Depth of Inheritance Tree*)**. La métrica DIT mide el máximo nivel en la jerarquía de herencia. Se trata de la cuenta directa de los niveles de la jerarquía de herencia, considerando que en el nivel cero de la jerarquía se encuentra la clase raíz. DIT se considera como una métrica del número de clases antecesoras que una clase podría potencialmente afectar, debido a que cuanto mayor sea el nivel de profundidad de herencia de una clase mayor es el número de métodos y atributos que hereda de otras clases.
- **Número de Hijos (NOC, *Number of Children*)**. NOC es el número de subclases subordinadas a una clase en la jerarquía, es decir, la cantidad de subclases que pertenecen a una clase. Según (Chidamber and Kemerer 1994), NOC es un indicador del nivel de reutilización, de la posibilidad de haber creado abstracciones erróneas, y del nivel de pruebas requerido.

Como ejemplo de cálculo de las métricas DIT y NOC, considérese el diagrama de UML de la Figura 4.20. Los valores de la métrica DIT serían: $DIT(\text{Clase Persona})=0$ (clase persona es la clase raíz o nivel 0), $\text{Clase Cliente}=1$, $\text{Clase Empleado}=1$, $\text{Clase Empleado Fijo}=2$, y $\text{Clase Empleado Temporal}=2$. En cuanto a la métrica NOC, los valores resultantes serían: $NOC(\text{Clase Persona})=2$, $\text{Clase Cliente}=0$ y $\text{Clase Empleado}=2$.

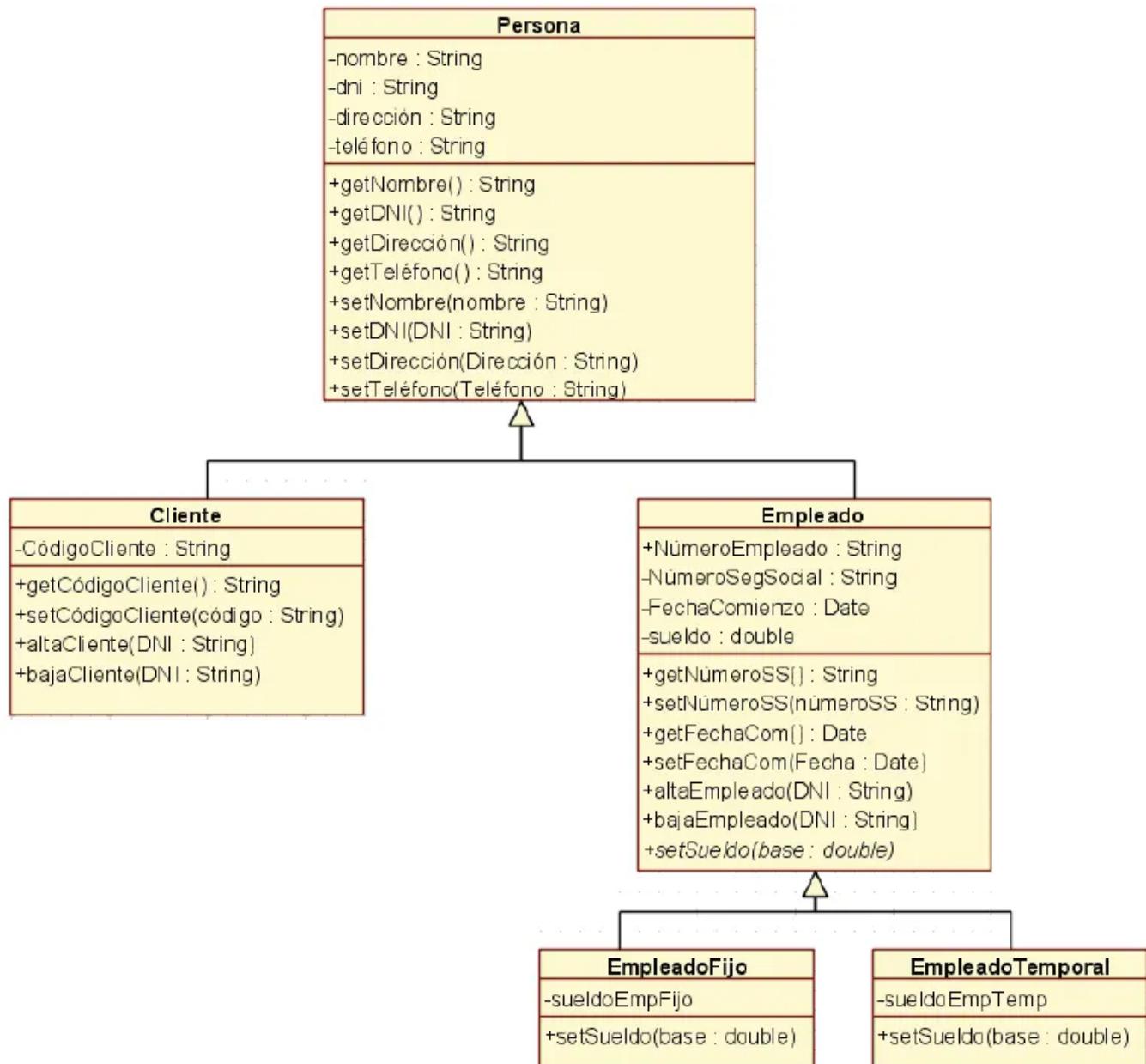


Figura 4.20 Diagrama de clases UML de ejemplo

- **Acoplamiento entre Objetos (CBO, Coupling Between Objects).** La métrica CBO indica para una clase el número de otras clases con las que está acoplada. Se considera que un objeto está acoplado a otro cuando actúa sobre ese otro objeto, por ejemplo cuando un método de un objeto utiliza un método de otro objeto. Esta métrica se considera útil para predecir el esfuerzo necesario para el mantenimiento y las pruebas. Para ilustrar el cálculo de la métrica CBO considérese el siguiente ejemplo ilustrado con un diagrama de clases UML en la Figura 4.21.

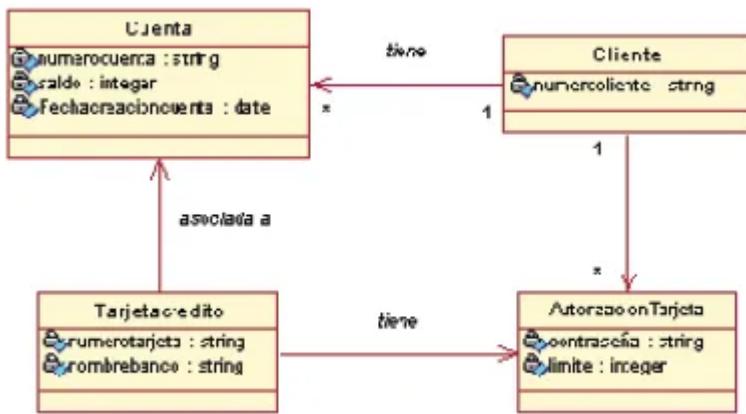


Figura 4.21 Diagrama de ejemplo para el cálculo de CBO

El acoplamiento entre objetos para cada clase sería: CBO (Cuenta) = 0, CBO (Tarjeta Crédito)= 2 (Usa métodos de las clases Cuenta y Autorización de Tarjeta) y CBO (Cliente)=2 (Usa métodos de las clases Cuenta y Autorización de Tarjeta).

- **Respuesta de una clase (RFC, Response For a Class).** RFC indica el número de métodos que pueden ser ejecutados potencialmente como respuesta a un mensaje recibido por un objeto de esa clase. RFC por lo tanto se calcula contando las ocurrencias de llamadas a otras clases de una clase particular. La

fórmula para calcular esta métrica es la siguiente: $RFC = |RS|$, donde RS es el conjunto respuesta para la clase. El conjunto respuesta para la clase se puede expresar de la siguiente manera: $RS = \{\{M\} \cup \{\text{all } i\} \setminus \{R_i\}\}$, donde $\{R_i\}$ es el conjunto de métodos llamados por el método i y $\{M\}$ es el conjunto de todos los métodos de la clase.

Para una mayor comprensión del modo de cálculo de RFC, considérese que se tiene un sistema compuesto por tres clases A, B y C, de modo que la clase A tiene 4 métodos A::f₁, f₂, f₃, f₄; la B tiene 4 métodos B::f₁, f₂, f₃, f₄ y la clase C tiene 5 métodos C::f_{1..f5}. Las invocaciones de los métodos de A son según este esquema:

Clase A con cuatro métodos:

A::f₁() invoca B::f₁(), B::f₂() y C::f₃()

A::f₂() invoca B::f₁()

A::f₃() invoca A::f₄(), B::f₃(), C::f₁() y C::f₂()

A::f₄() No llama a otros métodos

Entonces

$RS = \{A::f_1, A::f_2, A::f_3, A::f_4\}$

$\cup \{B::f_1, B::f_2, C::f_3\}$

$\cup \{B::f_1\}$

$\cup \{A::f_4, B::f_3, C::f_1, C::f_2\}$

= {A::f₁, A::f₂, A::f₃, A::f₄, B::f₁, B::f₂, B::f₃,

C::f₁, C::f₂, C::f₃}

Resultando $RFC(A)=10$

La métrica RFC se ha calculado para la clase A, la cual tiene 4 métodos locales y llama desde esos métodos a 6 métodos remotos (B: f₁, f₂, f₃ y C:f_{1,f2,f3}). La métrica RFC (Respuesta para una clase) se considera por tanto como la suma de los métodos locales a una clase, más los métodos remotos (métodos invocados de otras clases desde los métodos locales a la clase)

- **Falta de cohesión en los métodos (LCOM, Lack of Cohesion in Methods).** LCOM establece en qué medida los métodos hacen referencia a los atributos. Se calcula como el número de pares de funciones sin variables compartidas de instancia menos el número de pares de funciones con variables de instancia compartidas. LCOM es una métrica de la cohesión de una clase en base al número de atributos comunes usados por diferentes métodos. Un valor alto en LCOM implica falta de cohesión, es decir, escasa similitud entre los métodos siendo siempre deseable un alto grado de cohesión en los métodos de una clase.

Para ilustrar el cálculo de LCOM, considérese la Figura 4.22, donde los óvalos representan los métodos de una clase y los puntos representan los atributos de la clase. Un punto estará dentro de un óvalo perteneciente a un método, si en el mismo se hace referencia al atributo representado por el punto.

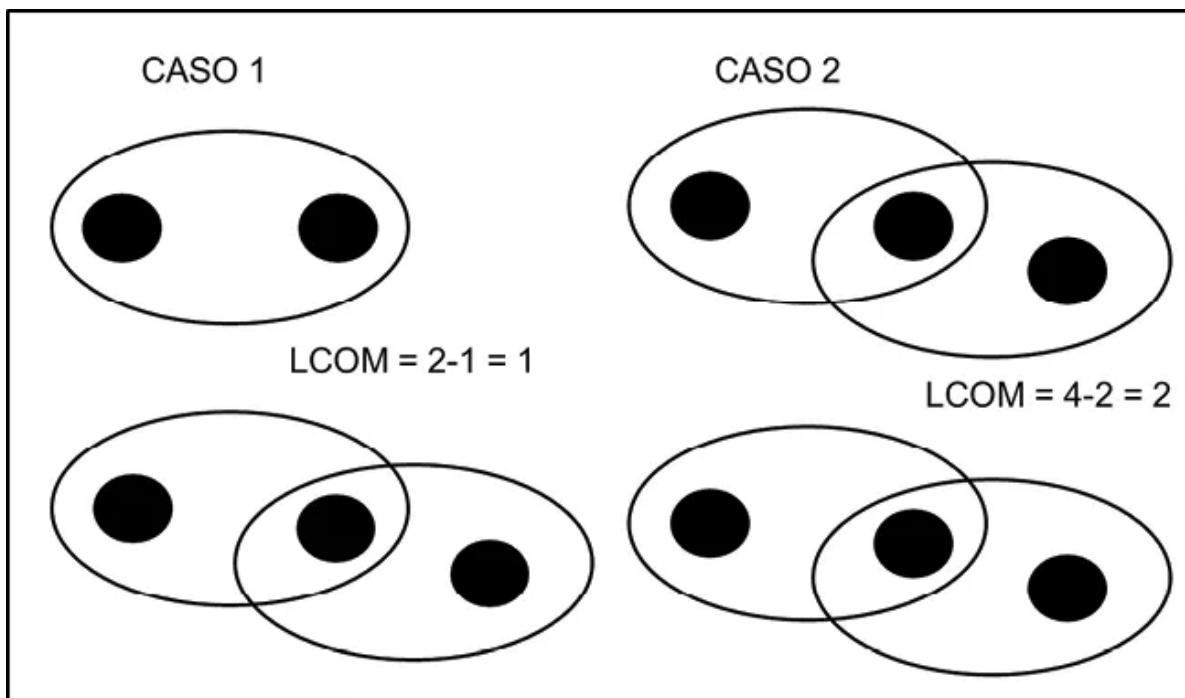


Figura 4.22 Diagrama de ejemplo para el cálculo de LCOM y LCOM*

En la Figura 4.22 se ve como LCOM se incrementa según se va incrementando el número de eslabones en la cadena. Esto no es deseable, ya que LCOM mide la cohesión y no debe depender del número de métodos en la cadena, sino de en qué medida los métodos afectan a los atributos de la clase. En el ejemplo de la 17.23 con 3 métodos en la cadena LCOM=0. En cambio, con 5 métodos en la cadena LCOM=2. Debido a estos problemas (Henderson-Sellers, 1996) propone LCOM* como medida de cohesión, que se calcula mediante la siguiente fórmula, siendo $M(A_i)$ = número de métodos que accede al atributo i ; m = número de métodos de la clase.

$$LCOM^* = \frac{PROMEDIO (\forall_i M(A_i)) - |m|}{|1 - m|}$$

En el caso 1 de la Figura 4.23 el cálculo de esta métrica sería: $i=6$ (n^o atributos), $m=3$ (n^o métodos), $M(A_1)=M(A_2)=M(A_3)=M(A_4)=M(A_6)=1$, $M(A_5)=2$, $LCOM^* = [1/6 * (1+1+1+1+2+1)] - 3 / (1-3)$, siendo $LCOM^*=0'916$.

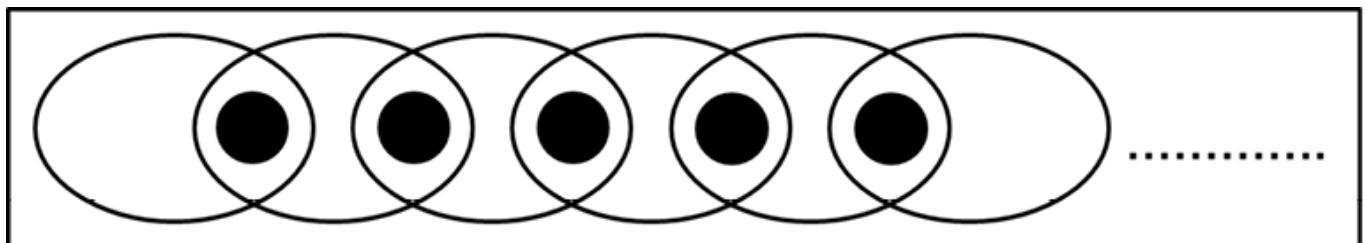


Figura 4.23 Incremento de LCOM

En Arvanitou et al. (2017) se presenta un mapeo sistemático que señala que la mantenibilidad es la característica (atributo de calidad) más estudiada, que normalmente la evaluación de las características de calidad se lleva a cabo por una sola métrica, en lugar de una combinación de varias, y resume las principales métricas utilizadas (con su frecuencia) para diferentes características de calidad como se muestra en la Tabla 4.5.

Atributo de calidad	Métrica de calidad	Frec.
Mantenibilidad	Depth of Inheritance Tree (DIT)	6
	Lines of Code (LOC)	6
	Weighted Methods per Class (WMC)	6
	Cyclomatic Complexity (CC-VG)	5
	Lack of Cohesion of Methods-1 (LCOM1)	5
	Tight Class Cohesion (TCC)	4
	Number of Children (NOC)	4
	Response for Class (RFC)	4
	Message Passing Coupling (MPC)	4
	Data Abstraction Coupling (DAC)	4
Reusabilidad	Number of Methods (NOM)	4
	Lack of Cohesion of Methods-1 (LCOM1)	3
	Lines of Code (LOC)	2
	Coupling Between Objects (CBO)	2
	Response for Class (RFC)	2
	Message Passing Coupling (MPC)	2
	Weighted Methods per Class (WMC)	2
	Number of Children (NOC)	2
	External Class Complexity (ECC)	2
	External Class Size (ECS)	2

Propensión al cambio	Depth of Inheritance Tree (DIT)	3
	Number of Children (NOC)	3
	Coupling Between Objects (CBO)	3
	Response for Class (RFC)	3
	Lack of Cohesion of Methods-1 (LCOM1)	3
	Data Abstraction Coupling (DAC)	3
	Number of Attributes (NA)	3
Comprendibilidad	Lines of Code (LOC)	3
	Depth of Inheritance Tree (DIT)	2
	External Class Complexity (ECC)	2
	External Class Size (ECS)	2
Facilidad de prueba <i>(testability)</i>	Response for Class (RFC)	4
	Coupling Between Objects (CBO)	3
	Lack of Cohesion of Methods-1 (LCOM1)	3
	Lines of Code (LOC)	3
Modificabilidad	Depth of Inheritance Tree (DIT)	2
	Halstead n1	2
	Halstead n2	2
Estabilidad	Weighted Methods per Class (WMC)	2
	Lines of Code (LOC)	2
	System Design Stability (SDI)	2

Tabla 4.5. Características y métricas de calidad

Se destaca también que las métricas que mayor validez empírica presentan son DIT, NOC, RFC, LCOM1 y WMC

Por su parte, Núñez-Varela et al. (2017) realizaron un mapeo sistemático sobre métricas para código fuente, de modo que

seleccionaron como relevantes un total de 226 artículos, publicados entre 2010 y 2015. Los trabajos sobre métricas de programación orientada a objetos, abarca más del 85% de los trabajos. En la tabla 4.6 se muestran las métricas, así como sus ocurrencias en los trabajos seleccionados en el estudio.

Métricas (Orientación a Objetos)	Ocurrencias
Weighted Methods per Class (WMC)	89
Coupling Between Objects (CBO)	89
Lack of Cohesion in Methods (LCOM)	86
Depth of Inheritance Tree (DIT)	81
Lines of Code (LOC)	79
Number of Children (NOC)	77
Response for a Class (RFC)	72
Number of Methods (NOM)	57
Cyclomatic Complexity (V(G))	55
Number of Attributes (NOA)	43
Fan-out (FANOUT)	27
Fan-in (FANIN), Number of Public Methods (NOPM)	22
Lines of Comments (LCOMM)	21
Afferent Couplings (Ca), Efferent Couplings (Ce)	20
Lack of Cohesion in Methods 2 (LCOM2)	18
Lines of Source Code (SLOC)	17
Lack of Cohesion in Methods 3 (LCOM3)	16
Cohesion Among Methods (CAM), Number of Classes (NCLASS), Number of Parameters (NPAR)	15
Nesting level (NEST), Message Passage Coupling (MPC)	14
Number of Overriden Methods (NMO), Number of Public Attributes	13

(NOPA)	
Lack of Cohesion in Methods 4 (LCOM4)	12
Effort (E), Instability (I), Lack of Cohesion in Methods 5 (LCOM5), Tight Class Cohesion (TCC)	11
Abstractness (A), Loose Class Cohesion (LCC), Numer of Statements (STAT), Number of Methods Inherited (NMI)	10
Attribute Hiding Factor (AHF), Class Cohesion (CC), Data Abstraction Coupling (DAC), Data Access Metric (DAM), Method of Aggregation (MOA), Method Hiding Factor (MHF), Normalized Distance from Main Sequence (Dn)	9

Tabla 4.6. Métricas OO y ocurrencias (Núñez-Varela et al., 2017)

LECTURAS RECOMENDADAS

- *Genero, M., Piattini, M. y Calero, C. (Eds.). 2005. Metrics for Software Conceptual Models. Imperial College Press.*
En esta recopilación se incluyen las propuestas más importantes sobre las métricas para modelos conceptuales.
- *Jones, C. (2017). A Guide to Selecting Software Measures and Metrics. CRC Press.*
En este libro se dan consejos muy relevantes sobre buenas prácticas a la hora de seleccionar y utilizar las métricas de software.
- *Rodríguez, M. y Piattini, M. (2020). Calidad del Software. Evaluación y Certificación de Productos Software, Madrid, Ra-Ma.*
Este libro profundiza en los modelos de calidad y métricas para el producto software, especialmente las relativas a la mantenibilidad y la adecuación funcional.

CONTROL ESTADÍSTICO DE PROCESOS

El control estadístico de procesos (*Statistical Process Control*, SPC) tiene su origen a comienzos del siglo XX, más concretamente a principios de los años 20, cuando se inicia la introducción de técnicas estadísticas para la inspección final de productos. Fue propuesto en 1924 por Walter Shewhart, que desarrolló el concepto estadístico de gráficos de control y proceso bajo control estadístico.

El SPC comenzó a utilizarse ampliamente en el sector industrial con el objetivo de controlar y mejorar los procesos de fabricación. Pero es durante la Segunda Guerra Mundial cuando, con más intensidad, se introdujeron dichas técnicas en la fabricación de armamento bélico, con el fin de disminuir el número de fallos del material durante los combates.

En este capítulo se presenta la aplicación de SPC en el campo de los procesos software, abordando en primer lugar los conceptos básicos y los gráficos de control utilizados para posteriormente describir las particularidades de los procesos software que hacen necesario adaptar adecuadamente dicha aplicación. Finalmente se muestran a modo de

ejemplo algunos trabajos representativos de aplicación de SPC en software.

FUNDAMENTOS DE SPC

SPC se aplica para analizar las **variaciones** de un proceso de forma que se pueda determinar si es estable o no. Todo proceso, por naturaleza, puede presentar variaciones. Tal como se aprecia en el modelo básico de proceso de la Figura 5.1, las variaciones pueden aparecer en la entrada, en el proceso y en las salidas. Toda variación en la entrada produce variación en la salida y cualquier variación del proceso también puede producir variación en la salida. Si esta variación en la salida es muy significativa entonces el producto o servicio no satisface los requisitos del cliente/usuario. Por ello se trata de eliminar cualquier variación significativa que reduzca notablemente la calidad esperada del producto o servicio.



Figura 5.1 Variaciones en un proceso

De las variaciones que pueden aparecer en un proceso es importante distinguir dos tipos (Figura 5.2) (Russel y Taylor, 2006):

- **Variación** debida a **causas comunes**, que son variaciones aleatorias que están siempre presentes en un proceso y que no se pueden identificar. Este tipo de variación es pequeño.

- **Variación** debida a causa **asignable**, que son variaciones no aleatorias que suelen ser debidas a cambios en las entradas del proceso o al propio proceso (como por ejemplo un trabajador que está enfermo o una máquina que se ha averiado). Este tipo de variación es la que puede y debe ser identificada ya que su efecto es mucho mayor.

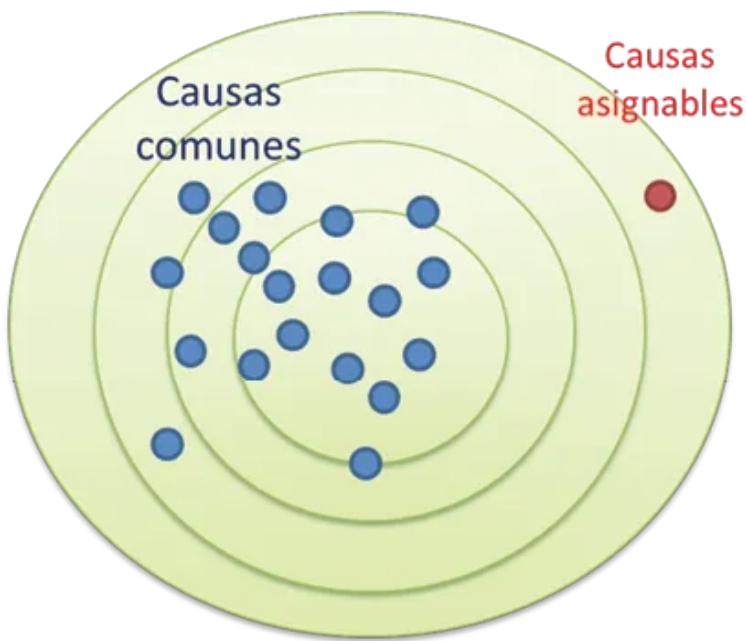


Figura 5.2 Causas comunes vs. Causas asignables

Por tanto, mediante SPC se trata de analizar un proceso para detectar la presencia de causas asignables de modo que se puedan disponer los medios necesarios para detectar la causa o causas y corregirlas.

El SPC se aplica sobre ciertas **características** o **medidas** del proceso (como pueden ser defectos, satisfacción del cliente, productividad, etc.). Se distinguen dos tipos:

- **Atributo**: puede ser evaluada con una respuesta discreta (por ejemplo, bueno-malo, sí-no); escala en una encuesta de satisfacción; número entero (por ejemplo, número de defectos).

- **Variable:** es un resultado expresado en una escala no discreta, como por ejemplo un valor expresado en números reales (peso, longitud, etc.).

En SPC para la evaluación de un proceso se evalúa una serie de muestras, siendo una **muestra** un subconjunto de elementos seleccionados para inspección.

Proceso de aplicación de SPC

El proceso SPC (ver Figura 5.3) comienza con la selección del proceso a evaluar, por ejemplo, el proceso de inspección de una empresa desarrolladora de software, así como de las características de rendimiento del proceso bajo evaluación. Por ejemplo, podría ser de interés analizar la eficiencia del proceso de inspección evaluada como el número de defectos detectados al día. El siguiente paso consiste en seleccionar el gráfico de control más apropiado, tal como se describe con más detalle en el siguiente sub-apartado. Los datos de las medidas de rendimiento del proceso deben ser recopilados durante largos períodos de tiempo para que el análisis SPC sea significativo, tras lo cual, en base a dichos datos se calcula la línea central y los límites del gráfico de control y se representan dichos datos relevantes en el gráfico. Una vez representados se debe aplicar una serie de pruebas o tests para determinar si el proceso está bajo control estadístico. En caso de no obtenerse fallo en los test, se dice que el proceso es estable y se debe seguir monitorizando. En caso contrario, se debe buscar la causa o causas que provocan el fallo, eliminar dichas causas y se vuelve al paso de recopilación de mediciones del proceso para evaluar si está bajo control.

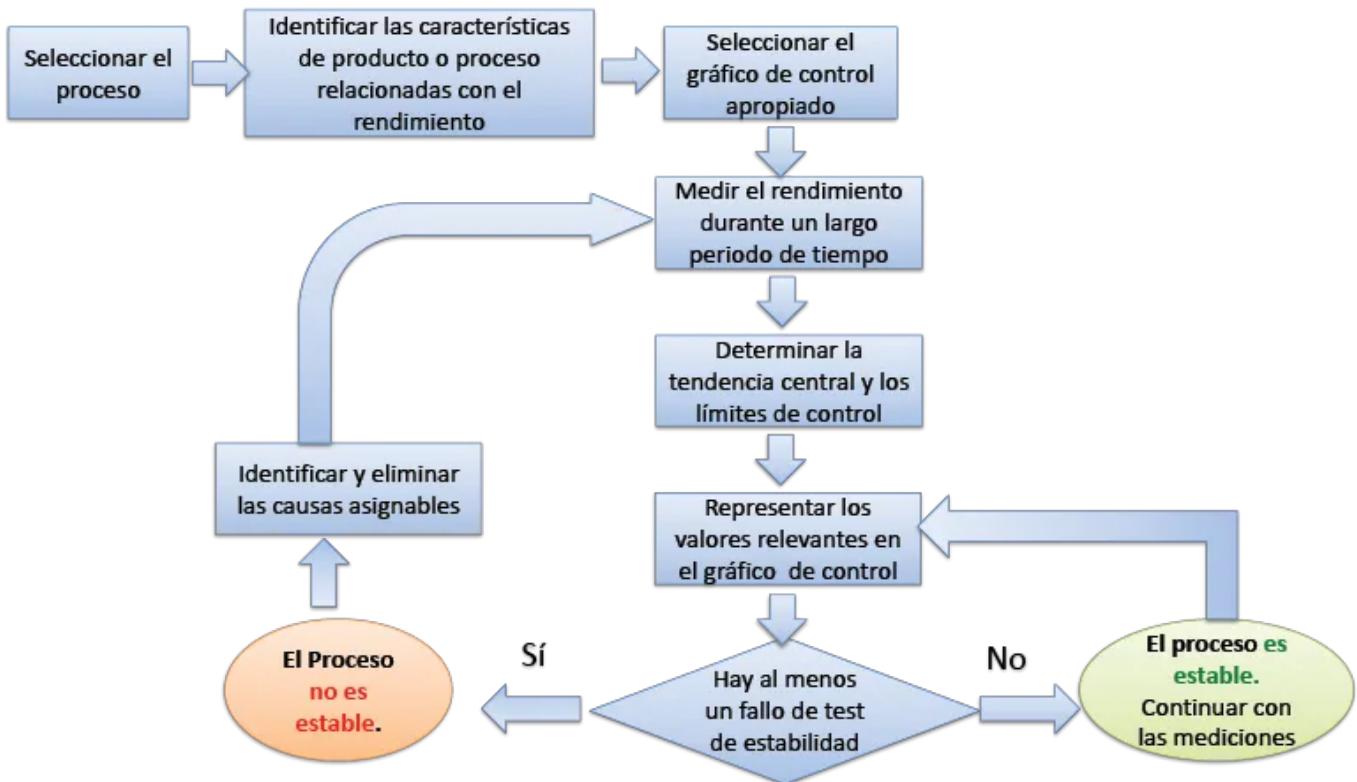


Figura 5.3 Proceso SPC

Gráficos de control en SPC

Los gráficos de control en SPC se utilizan para identificar de forma visual la variación de los procesos a lo largo de tiempo, de modo que sirvan como punto de partida para identificar variaciones debidas a causas asignables y buscar dichas causas. Representan por tanto si los datos de una muestra están dentro o fuera del rango normal de variación. Un proceso está “fuera de control” si una o más muestras caen fuera de sus límites de control. En la Figura 5.4 se muestra un ejemplo típico de gráfico de control en el que tal como se puede observar se representa en el eje Y la característica del proceso a evaluar (mediciones de la característica), en el eje X la muestra u observación, y cada punto representa el valor de la característica para cada observación. Mediante líneas horizontales se representa la tendencia

central (CL, *Center Line*) y los límites superiores (UCL, *Upper Control Limit*) e inferior (LCL, *Lower Control Limit*).

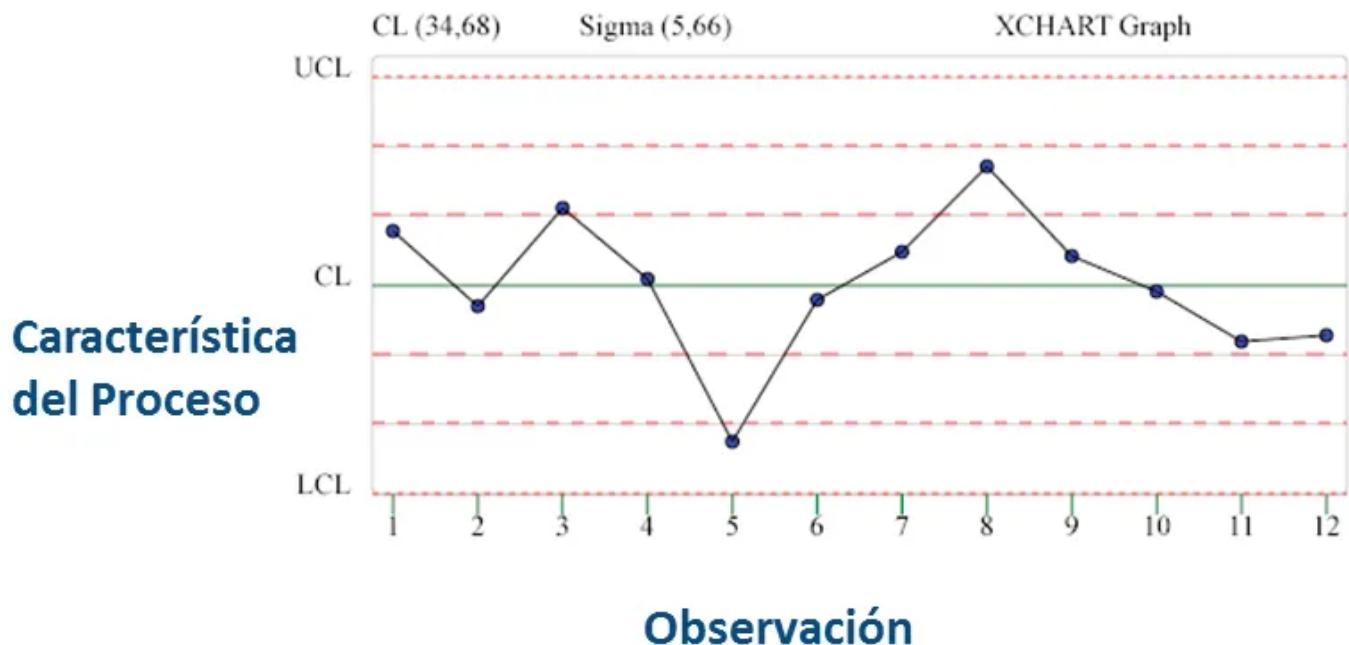


Figura 5.4 Ejemplo de gráfico de control

La fórmula general para calcular la tendencia central y los límites es la siguiente:

- $UCL = \mu + z\sigma$
- $CL = \mu$
- $LCL = \mu - z\sigma$

μ representa la media y σ la desviación típica de la variable a evaluar, mientras que z es la distancia de los límites de control a la línea central, expresado en términos de unidades de desviación típica. Tal como se ha descrito previamente, el valor de $z=3$ es el tradicionalmente aceptado como estándar en la industria (que supone *seis sigmas* de separación).

Para la realización de SPC se dispone de un amplio conjunto de gráficos de control en la bibliografía, entre los cuales debemos distinguir

fundamentalmente entre dos tipos:

- **Gráficos de control para variables**, como por ejemplo los gráficos X y R que se aplican a datos que se miden en una escala continua, como por ejemplo resultados expresados en números reales (ejemplos: volumen, altura, peso, etc.).
- **Gráficos de control para atributos**, que evalúan resultados discretos, como los expresados como números enteros (número de quejas), valores que indican la presencia/ausencia de algo, etc. Ejemplo de gráficos para atributos son los gráficos P, nP para proporciones (ejemplo, proporción de botellas rotas en un lote) y C o U para evaluar número de elementos (número de quejas de clientes en un mes).

A continuación, se resumen las principales características de algunos de los gráficos de control más representativos:

- **Gráfico X-bar (X barra) y R-Bar.** Permiten monitorizar un proceso a lo largo del tiempo en base a la media de valores (X-bar) o del rango (R-bar) de una serie de muestras, que se denominan subgrupos. Por tanto, se usan cuando se pueden recoger varias observaciones por muestra (entre 2 y 10). Cada subgrupo representa como una instantánea del proceso en un determinado momento de tiempo. Al ser gráficos basados en la representación temporal, las muestras deben visualizarse ordenadas por el tiempo (en el eje X). Si el tamaño de cada muestra es superior a 10 observaciones, se aconseja usar **gráficos S (sigma)**, ya que el rango es un estimador poco fiable en caso de subgrupos grandes. En este caso en vez de la media se usa la desviación estándar de las unidades u observaciones por muestra. Para subgrupos con una

observación se deben usar **gráficos individuales X y de rango (Moving Range, MR)**, entre otros, en los que cada punto de datos es un valor individual y no promedio.

- **Gráfico C.** Este gráfico usa una muestra de tamaño constante de datos de tipo atributo, donde el tamaño en promedio de la muestra es superior a 5. Con este gráfico se pretende determinar la estabilidad de datos habitualmente “contados”, como por ejemplo errores por característica, pedidos al mes, etc. Es de mayor utilidad, por ejemplo, cuando se conoce cuántos defectos hay y no sólo el número de unidades defectuosas, considerando que el tamaño de la muestra es siempre el mismo. Si el tamaño de la muestra es variable se usa el **gráfico U**. Para este tipo de gráficos lo apropiado es que la distribución de los datos sea Poisson.
- **Gráfico p.** Este gráfico usa un tamaño de muestra habitualmente igual o mayor a 50. Se usa habitualmente para representar la proporción de unidades defectuosas en un grupo cuando el tamaño de la muestra varía. Por ejemplo, el número de botellas defectuosas en un lote. De modo que se trabaja con proporciones de unidades defectuosas por lote. Si el tamaño de la muestra es el mismo en todas, entonces se hace uso del **gráfico nP**. Por ejemplo, supongamos el número de comidas “defectuosas” en un restaurante. Para este tipo de gráficos lo apropiado es que la distribución de los datos sea binomial.

El proceso de selección del gráfico de control más apropiado es resumido en la Figura 5.5.

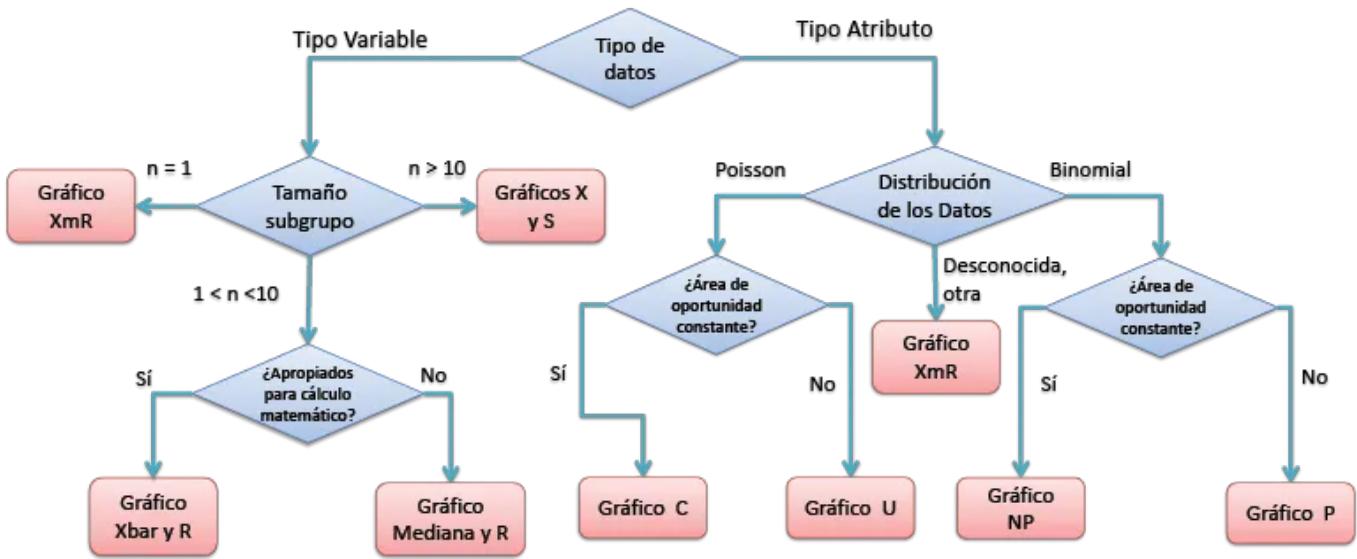


Figura 5.5 Decisión sobre tipo de gráfico

En la Tabla 5.1 se muestran las fórmulas para el cálculo de la línea central y los límites de los gráficos de control descritos con anterioridad.

Gráfico	Línea Central	Límite Superior	Límite Inferior
X-bar (desv. típica proceso conocida σ)	$\bar{x} = \frac{\sum_{m=1}^k x_m}{n}$	$UCL = \mu + \frac{3\sigma}{\sqrt{n}}$	$UCL = \mu - \frac{3\sigma}{\sqrt{n}}$
R-bar (desv. típica proceso conocida σ)	$\bar{R} = \frac{\sum_{i=1}^k R_i}{n}$	$UCL = \bar{R}D_4$	$LCL = \bar{R}D_3$
X-bar (desv. típica proceso desconocida)	$\bar{x} = \frac{\sum_{m=1}^k x_m}{n}$	$UCL = \bar{x} + A_2 \bar{R}$	$LCL = \bar{x} - A_2 \bar{R}$
R-bar (desv. típica proceso desconocida)	$\bar{R} = \frac{\sum_{i=1}^k R_i}{n}$	$UCL = \bar{R}D_4$	$LCL = B_3 \bar{s}$
X ($n=1$)	$\bar{x} = \frac{\sum_{m=1}^k x_m}{n}$	Véase XBar	Véase XBar
R ($n=1$)	$\bar{R} = \frac{x_{max} - x_{min}}{k}$	Véase RBar	Véase RBar
S-bar	$\bar{s} = \frac{\sum s_i}{k}$ $s_i = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n-1}}$	$UCL = B_4 \bar{s}$	$LCL = B_3 \bar{s}$

Gráfico P	$\bar{p} = \frac{n^o \text{ totalelemdef}}{n^o \text{ totalelem}}$	$UCL = \bar{p} + z\sqrt{\frac{\bar{p}(1-\bar{p})}{k}}$	$LCL = \max[0, \bar{p} - z\sqrt{\frac{\bar{p}(1-\bar{p})}{k}}]$
Gráfico nP	$\bar{p} = \frac{\sum_{m=1}^k d_m}{n}$	$UCL = n\bar{p} + z\sqrt{n\bar{p}(1-\bar{p})}$	$LCL = \max[0, n\bar{p} + z\sqrt{n\bar{p}(1-\bar{p})}]$
Gráfico C	$\bar{c} = \frac{n^o \text{ totaldefectos}}{k}$	$UCL = \bar{c} + z\sqrt{\bar{c}}$	$LCL = \bar{c} - z\sqrt{\bar{c}}$
Gráfico U	$\bar{u} = \frac{\sum_{m=1}^k u_m}{k}$ $u_m = \frac{n^o \text{ totaldefectosenla muestra}}{\text{tamaño muestra}}$	$UCL = \bar{u} + z\sqrt{\frac{\bar{u}}{k}}$	$LCL = \max[0, \bar{u} - z\sqrt{\frac{\bar{u}}{k}}]$

k = número de muestras
n = número de elementos/observaciones por muestra
z = distancia (número de sigmas)
c = número de defectos por muestra
d_m = número de defectuosos en muestra m

Tabla 5.1. Cálculo de la tendencia central y los límites en los gráficos de control

Por su parte, en la Tabla 5.2 se muestra un extracto de las constantes de los gráficos de control, que son obtenidas a partir de las distribuciones estadísticas en las que se basa cada tipo de gráfico:

	X-Bar		R		Estimación Sig		S
Tamaño muestra	A ₂	A ₃	D ₃	D ₄	d ₂	B ₃	B ₄
2	1.880	2.659	0	3.267	1.128	0	3.267
3	1.023	1.954	0	2.574	1.693	0	2.568
4	0.729	1.628	0	2.282	2.059	0	2.266
5	0.577	1.427	0	2.114	2.326	0	2.089
6	0.483	1.287	0	2.004	2.534	0.030	1.970
7	0.419	1.182	0.076	1.924	2.704	0.118	1.882
8	0.373	1.099	0.136	1.864	2.847	0.185	1.815

9	0.337	1.032	0.184	1.816	2.970	0.239	1.761
10	0.308	0.975	0.223	1.777	3.078	0.284	1.716
11	0.285	0.927	0.256	1.774	3.173	0.321	1.679
12	0.266	0.886	0.284	1.716	3.258	0.354	1.646
13	0.249	0.850	0.308	1.692	3.336	0.382	1.618
14	0.235	0.817	0.329	1.671	3.407	0.406	1.594
15	0.223	0.789	0.348	1.652	3.472	0.428	1.572
.....

Tabla 5.2. Constantes para los gráficos de control

EJEMPLO DE GRÁFICO X-BAR Y R-BAR

Consideremos el siguiente ejemplo para la realización de un gráfico de control X-bar para variables. Supongamos que una empresa de fabricación de refrescos realiza el control estadístico de su proceso de fabricación para lo cual mide la altura de las latas de refrescos de 200ml que son fabricadas en dicho proceso. En la Tabla 5.3 se muestran las 20 muestras seleccionadas, cada una de las cuales está formada por 4 unidades.

Muestra Altura lata refresco

Nº	1	2	3	4	Media	Rango
1	15,02	15,90	15,68	15,32	15,48	0,88
2	15,58	15,88	15,13	15,17	15,44	0,75
3	15,63	15,49	15,93	15,80	15,71	0,45
4	15,86	15,77	15,42	15,53	15,64	0,45
5	15,00	15,73	15,37	15,53	15,41	0,73
6	15,12	15,29	15,40	15,50	15,33	0,38

7	15,58	15,46	15,58	15,97	15,65	0,50
8	15,09	15,59	15,48	15,85	15,50	0,77
9	15,70	15,97	15,26	15,21	15,53	0,76
10	15,02	15,64	15,84	15,50	15,50	0,82
11	15,12	15,77	15,19	15,05	15,28	0,72
12	15,41	15,09	15,93	15,73	15,54	0,83
13	15,70	15,69	15,23	15,33	15,49	0,47
14	15,79	15,07	15,33	15,60	15,45	0,73
15	15,04	15,20	15,70	15,19	15,28	0,66
16	15,55	15,98	15,78	15,23	15,63	0,76
17	15,48	15,28	15,52	15,33	15,40	0,23
18	15,22	15,76	15,24	15,02	15,31	0,74
19	16,00	15,70	16,30	16,00	16,00	0,60
20	15,78	15,52	15,11	15,90	15,58	0,79

Tabla 5.3. Ejemplo gráfico X-bar: altura de lotes de refresco

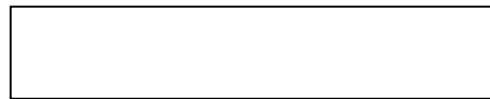
En la columna media de la Tabla 5.3 se indica la media del valor por muestra y la columna rango se obtiene calculando la diferencia entre el valor superior e inferior de la muestra. Para la obtención del gráfico de control, en primer lugar veamos cómo calcular los límites superior e inferior, así como la tendencia central. De acuerdo a las fórmulas de este tipo de gráfico, tendríamos:

$$\text{Tendencia central} = \text{Media de las medias} = (15,48 + 15,44 + 15,71 + 15,64 + 15,41 + 15,33 + 15,65 + 15,50 + 15,53 + 15,50 + 15,28 + 15,54 + 15,49 + 15,45 + 15,28 + 15,63 + 15,40 + 15,31 + 16,00 + 15,58) / 20 = \mathbf{15,51}$$

Si tenemos un conjunto de datos con desviación típica desconocida, debemos aplicar la fórmula correspondiente. En este caso, para obtener el valor de A2 se consultan las tablas de constantes de gráficos de

control X-bar, según las cuales **A₂=0,729**, para muestras de 4 elementos (véase Tabla 5.2). Por su parte, se obtiene el promedio de los rangos, cuyo resultado es:

$$\bar{R} = \frac{0,88 + 0,75 + \dots + 0,79}{20} = 0,65$$



límite superior e inferior del gráfico del siguiente modo:

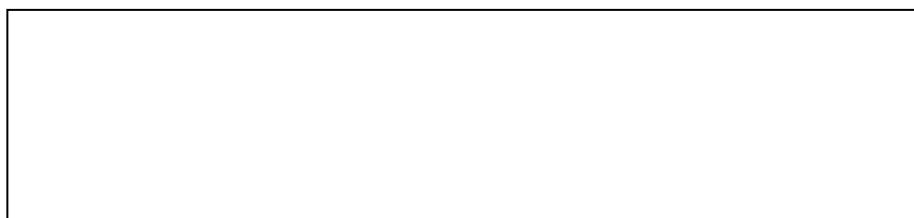


Figura 5.6.

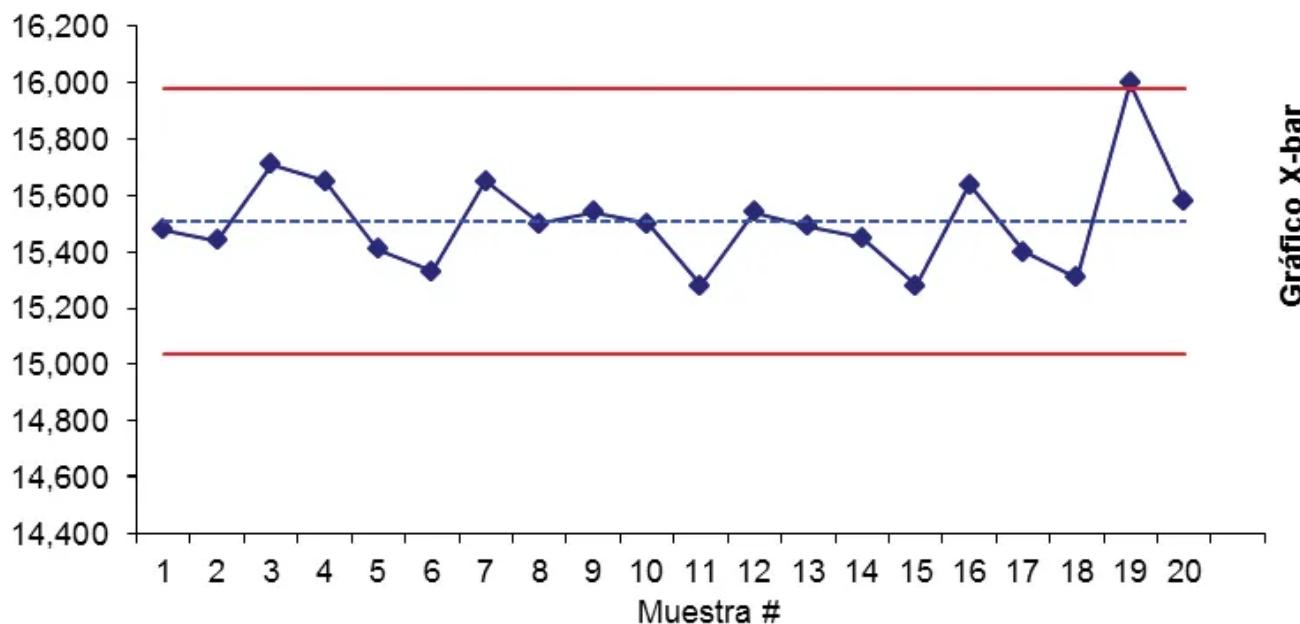


Figura 5.6 Gráfico de control X-bar del ejemplo

Tal como podemos observar en el gráfico de control, una de las observaciones (muestra 19) cae fuera de los límites de control por lo que habrá que analizar las causas de ello aplicando los criterios de decisión en este tipo de gráficos.

A continuación, veamos cómo calcular el gráfico R, ya que suele ser de mayor utilidad una interpretación conjunta de los gráficos X-bar y R, respecto a interpretaciones por separado. Para ello, el cálculo de los límites se realiza del siguiente modo:

$$LCL = D_3 \bar{R} = 0 * 0,65 = 0$$

$$UCL = D_4 \bar{R} = 2,282 * 0,65 = 1,48$$

En la Figura 5.7 se muestra el gráfico R resultante de lo anterior donde se puede apreciar que todas las observaciones están dentro de los límites de control. Ello refuerza la importancia de analizar conjuntamente los resultados tanto con gráficos X como R, dado que puede haber variaciones significativas en los promedios, pero no así en los rangos (como es el caso) o viceversa.

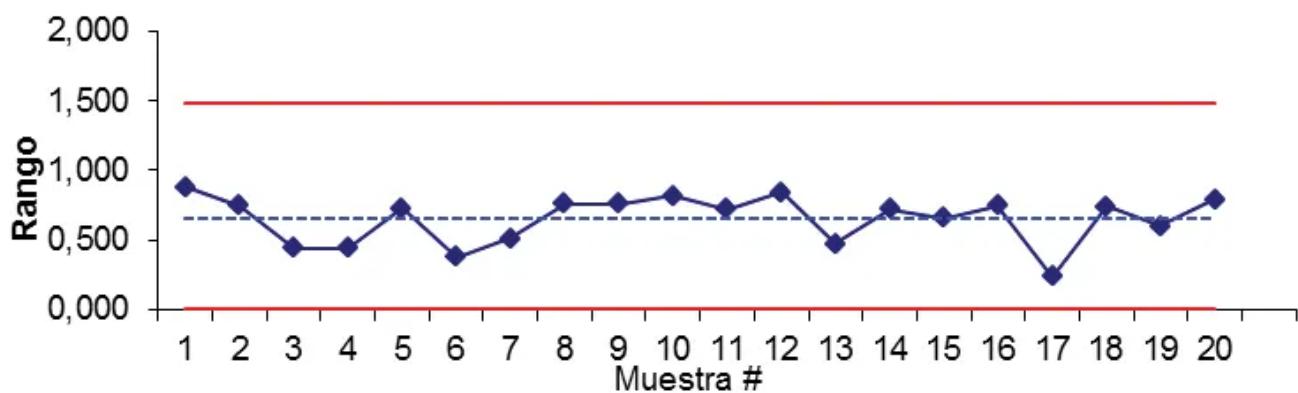


Figura 5.7 Gráfico de control X-bar del ejemplo

EJEMPLO DE GRÁFICO C

A continuación, se va a ilustrar el uso de gráficos C como ejemplo de gráfico de control para atributos. Supongamos que una empresa desarrolladora de software quiere analizar las incidencias de su producto software comercial para lo que evalúa las incidencias recibidas

por parte de sus clientes vía telefónica. En la Tabla 5.4 se muestra el número de incidencias recibidas por semana relativas a dicho software.

Semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Nº Incidencias	12	15	11	9	16	12	11	15	21	24	12	11	13	14	16

Tabla 5.4. Ejemplo gráfico C: Nº incidencias producto software

El límite central de este ejemplo será el promedio de incidencias por semana, que será el resultado de dividir 212 (suma total de incidencias) entre el número de semanas (15), obteniéndose:

$$\bar{C} = \frac{212}{15} = 14,13$$

Para este tipo de gráficos, los límites se calculan aplicando las siguientes fórmulas (véase Tabla 5.1), en las que “z” representa el número de sigmas a considerar (en este caso aplicaremos 3 sigmas como se realiza habitualmente en SPC):

$$LCL = \bar{c} - z\sqrt{\bar{c}} = 14,13 - 3 * \sqrt{14,13} = 2,85$$

$$UCL = \bar{c} + z\sqrt{\bar{c}} = 14,13 + 3 * \sqrt{14,13} = 25,41$$

El gráfico de control resultante se muestra en la Figura 5.8:

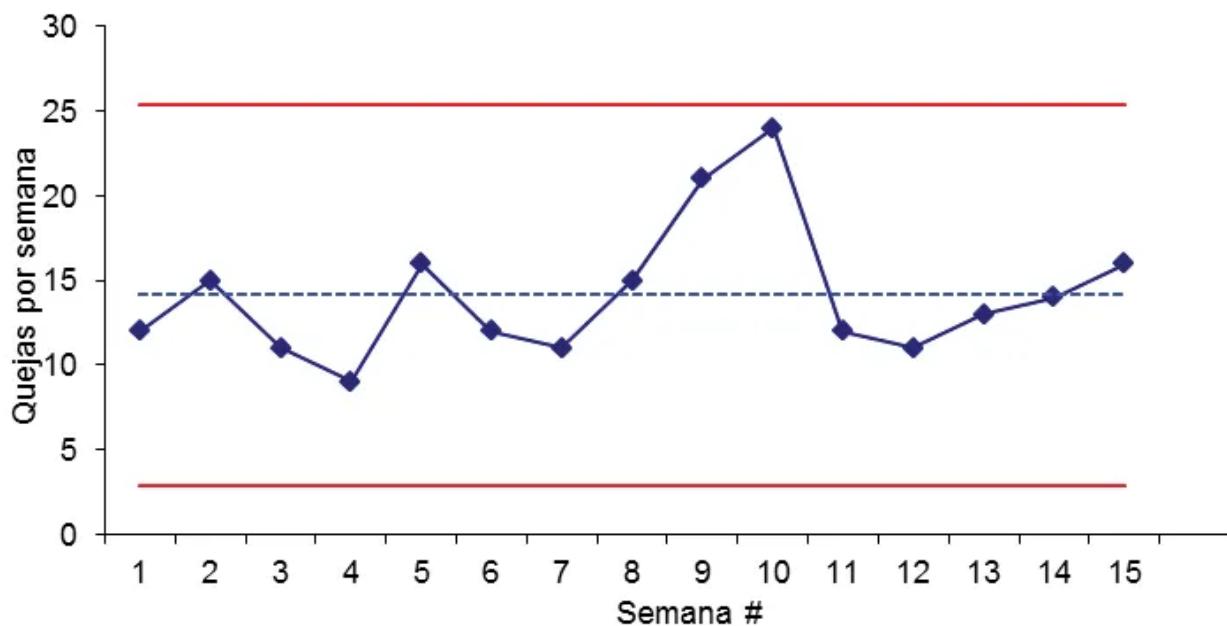


Figura 5.8 Gráfico de control X-bar del ejemplo

INTERPRETACIÓN DE LOS GRÁFICOS DE CONTROL

Una vez que se ha obtenido el gráfico de control es importante realizar una adecuada interpretación del mismo. Por lo general se distinguen varias zonas en el gráfico dependiendo de la distancia de cada dato al eje central, de modo que la zona A dista de la media más/menos tres veces la desviación típica y la zona B y C 2 y unas veces respectivamente (ver Figura 5.9) para interpretación en gráficos X-bar y R). En función de ciertos patrones en los datos se puede realizar una interpretación adecuada que conduzca a una mejor localización de las causas (en caso de proceso inestable).

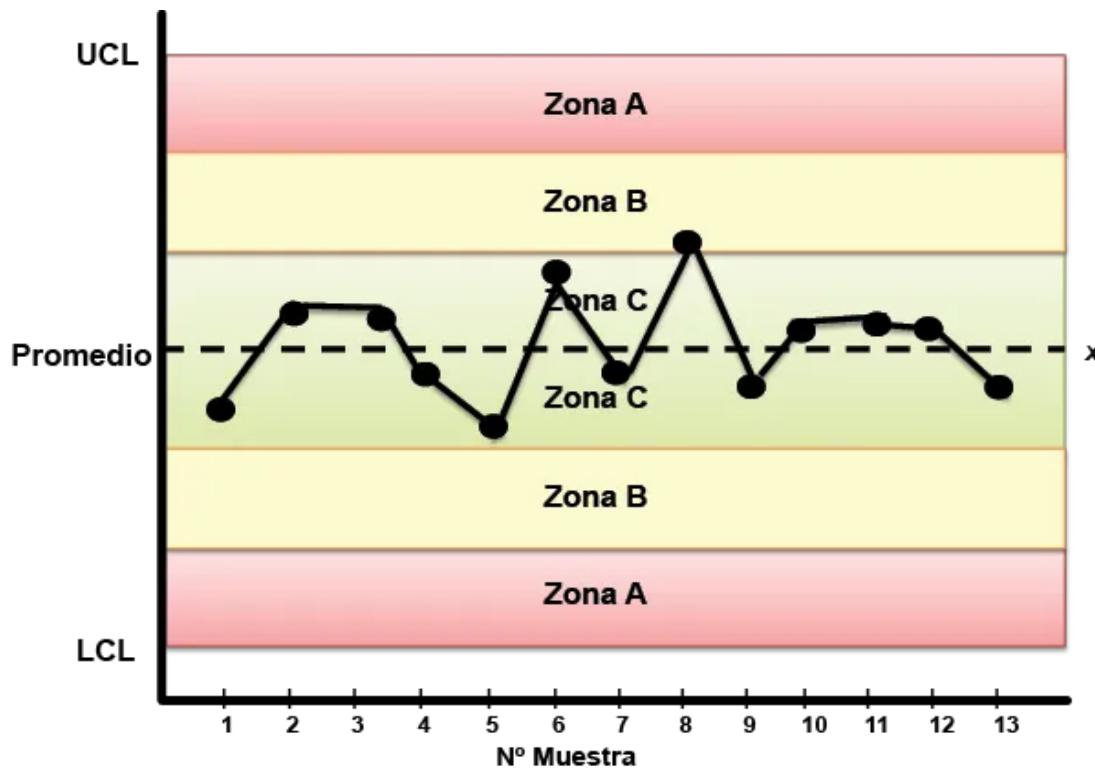


Figura 5.9 Interpretación de gráficos de control X-bar y R

Por ejemplo, algunas reglas tradicionalmente aceptadas son comprobadas en los siguientes *test*:

- **Test 1:** al menos un punto fuera de los límites de control (zona A).
- **Test 2:** al menos dos puntos de una secuencia de 3 que caen en un mismo lado o distan al menos dos sigmas respecto de la línea central y dos de tres puntos sucesivos en la zona A o más allá de ella.
- **Test 3:** hay al menos cuatro puntos de una secuencia de cinco en el mismo lado y distan al menos un sigma de la línea central o hay cuatro de cinco observaciones sucesivas en la zona B o más allá de ella.
- **Test 4:** hay al menos siete, ocho o nueve puntos que están en un mismo lado de la línea central.

- **Test 5:** hay por lo menos ocho observaciones sucesivas que están a los lados de la línea central, evitando la zona C.
- **Test 6:** hay al menos quince observaciones consecutivas en la zona C, por encima y/o por debajo de la línea central.
- **Test 7:** hay por lo menos catorce observaciones sucesivas que se alternan una por encima y otra por debajo respecto a la otra.
- **Test 8:** hay por lo menos seis observaciones sucesivas que forman una secuencia creciente o decreciente.

PROCESOS SOFTWARE VS. PROCESOS INDUSTRIALES

La aplicación del SPC en procesos software se viene investigando y aplicando desde los ochenta (Florac y Carleton, 1999), y ha tenido una gran repercusión en la industria del software promovida por los modelos de madurez CMM, CMMI, ISO/IEC 15504/33000, ya que es una técnica recomendada para el soporte a la gestión cuantitativa de procesos y mejora continua en niveles altos de madurez y por tanto viene siendo aplicado en las organizaciones con nivel 4 y 5 (Radice, 2000).

Por tanto, la utilización de SPC en la industria del software es relativamente joven y tiene el *hándicap* de que se han trasladado en su aplicación a la industria del software los mismos principios que se han utilizado en los procesos industriales de fabricación, no siendo directamente aplicables, ya que hay que tener en cuenta las diferencias entre los procesos software y los procesos de fabricación industrial (Lantz, 1992), mostradas en la Tabla 5.5.

Intensivos en Factor Humano, Actividad Cognitiva	Intensivos en Máquinas
Entradas y Salidas diferentes en cada ejecución	Entradas y Salidas iguales en cada ejecución
Alta variación en rendimiento del proceso debido al factor humano	Baja variación en rendimiento del proceso
Riesgos siempre presentes en todas las fases	La mayoría de los riesgos se concentran en la fase de diseño más que en producción
Es difícil de obtener conformidad con el producto en base a especificaciones de requisitos	Se puede comprobar la conformidad del producto respecto a las especificaciones de requisitos

Tabla 5.5. Procesos software vs. procesos industriales

El principal factor a considerar es que todo proceso software es intensivo en personas, es decir, el factor humano es fundamental y por tanto el proceso está dominado por actividades cognitivas. Cada entrada y salida de un proceso software es distinta en cada ejecución del mismo. El predominante factor humano implica diferencias en el rendimiento del proceso en cada ejecución y a este fenómeno se le denomina diversidad del proceso (conocido por sus siglas en inglés PS –Process Diversity–) (Lindvall y Rus, 2000; Baldassarre et al., 2004), que implica dificultad a la hora de predecir, monitorear y mejorar un proceso software.

Las diferencias entre procesos software y de fabricación comienzan desde el inicio del ciclo de vida, ya que la captura y el análisis de requisitos software suponen una actividad muy subjetiva en los que la habilidad, creatividad de los participantes humanos suponen mucha más variabilidad en comparación al trabajo de una máquina. Los procesos software no están por tanto automatizados al mismo nivel que los procesos de fabricación, en los que el grado puede llegar incluso al

100%. Por ello, en cada ejecución de un proceso software el riesgo de cometer errores está siempre presente y no puede ser eliminado. En los procesos de fabricación, una vez terminado un proyecto y probada la producción por primera vez, si todo está bien, es probable que lo esté para cada repetición. Otra diferencia importante es la influencia del desempeño de cada subprocesso individual. Un programador puede variar su productividad por muchas razones, desde que un dispositivo no funciona, un problema de salud, poco estado de ánimo, etc.

Cualquier cambio provoca inevitablemente una mayor desestabilización y un proceso menos predecible, como un cambio en el lenguaje de programación, que puede suponer una disminución drástica de la productividad hasta que con la formación y el uso se adquiere la habilidad necesaria para estabilizar dicha productividad. En la fabricación industrial este efecto no es el mismo, por ejemplo, cuando se introduce una nueva máquina en una línea de producción lo más probable es que funcione correctamente al inicio o poco después. Otro aspecto diferencial importante es la mayor facilidad con la cual se puede verificar la conformidad del producto final con los requisitos en un entorno de fabricación industrial en comparación al software.

En la Tabla 5.6, se resumen los aspectos especiales y dificultades a considerar en la aplicación de SPC en el desarrollo software de acuerdo a la revisión bibliográfica realizada por Chang y Tong (2013):

Dificultades de aplicación SPC en software	Fuente
<ul style="list-style-type: none">• Demasiados atributos y variables a considerar en el ciclo de vida del software	Baldassarre et al. (2004)
<ul style="list-style-type: none">• Dificultad a la hora de elegir indicadores adecuados para implementar SPC en procesos de desarrollo software	Caivano (2005)

<ul style="list-style-type: none"> • Especiales características de los procesos software (intensivos en factor humano y centrado en proceso) • Múltiples causas comunes de variación en procesos software • Dificultades para obtener un amplio conjunto de datos homogéneos 	Komuro (2006)
<ul style="list-style-type: none"> • El software lo producen las personas, no las máquinas • Demasiadas fuentes de variabilidad en el entorno de desarrollo software • Múltiples causas comunes de variación en procesos software • El indicador de comportamiento de procesos varía con el ciclo de desarrollo 	Manlove y Kan (2007)
<ul style="list-style-type: none"> • Fuentes de variación heterogéneas • Demasiados pocos puntos de variación existen a la hora de construir límites de control válidos • Se deben mantener demasiados gráficos de control al desagregar los datos según el atributo que caracterizan 	Raczynski y Curtis (2008)
<ul style="list-style-type: none"> • Múltiples causas comunes de variación en software • Dificultades en selección de métricas para implementar SPC • Dificultad en encontrar equilibrio entre número de puntos de datos y profundidad del análisis 	Sargut y Demirörs (2006)
<ul style="list-style-type: none"> • Muestreo racional de datos del proceso • Dificultad a la hora de elegir indicadores adecuados para implementar SPC en software • Carencia de datos suficientes 	Tarhan y Demirörs (2006)
<ul style="list-style-type: none"> • Pocos datos para construir gráficos de control • Múltiples causas comunes de variación en software 	Weller y Card (2008)

Tabla 5.6. Dificultades en la aplicación de SPC en procesos software (Chang y Tong, 2013)

En definitiva, se debe prestar especial atención a la aplicación del SPC en procesos software debido a su naturaleza con diferencias tan significativas respecto a los procesos industriales.

Para que las empresas desarrolladoras de software puedan aspirar a niveles altos de madurez, necesitan aplicar técnicas adecuadas para el análisis de grandes cantidades de datos que les permitan estabilizar los resultados de sus procesos y mantenerlos bajo control. Por ello hay que tener muy presente que tanto la cantidad como la calidad de los datos son muy importantes para la aplicación de SPC en el software, por lo que es necesario que las organizaciones dispongan de definición de indicadores de sus procesos de software y hagan una adecuada recopilación de datos.

CASOS DE APLICACIÓN DE SPC EN PROCESOS SOFTWARE

En este apartado se describen algunos casos representativos de aplicación de SPC en distintos procesos de software. Los procesos de desarrollo y revisión de código y de pruebas suelen ser los que más se han abordado, ya que son procesos para los que se dispone de un mayor conjunto de datos que se recopilan durante largos períodos de tiempo, aspecto fundamental para la monitorización del proceso mediante el uso de gráficos de control. También se ha aplicado SPC a otros procesos relevantes de desarrollo, como diseño y a procesos auxiliares del ciclo de vida, como los relacionados con la gestión de los proyectos software (análisis y estimación de recursos), mantenimiento y mejora. Estos resultados se confirman con la revisión sistemática realizada por Brito et al. (2018), en la que se identifican las métricas más comúnmente

utilizadas en análisis SPC en software, clasificadas en los procesos de reclutamiento, gestión y desarrollo de requisitos, mantenimiento, codificación, diseño, testing, gestión de proyectos, revisiones y corrección de defectos, inspección, calidad, entrega al cliente.

En relación a las propuestas para el uso de SPC en las organizaciones software, podemos encontrar en la bibliografía desde las propuestas tradicionales de Florac y Carleton (1999) y Pandian (2004), que establecen sus fundamentos de aplicación, a propuestas más recientes como las de Chang y Tong (2013), Lee y Park (2012) y Barcellos et al. (2013), que proporcionan guías adicionales para adaptar la aplicación tradicional de SPC a la problemática de la industria del software. A continuación, se muestran a modo de ejemplo algunas de las propuestas más representativas.

SPC: Proceso de Gestión del Proyecto

Jalote (1999), libro de experiencias de aplicación de CMM en la empresa Infosys, evalúa con SPC diversos procesos con gráficos de control, entre ellos el proceso de planificación y gestión del proyecto considerando la fiabilidad de la planificación de tiempo, esfuerzo y calidad. Para ilustrarlo, se realizan gráficos de control X para monitorizar la variación de calendario (ver Figura 5.10) y esfuerzo (ver Figura 5.11) y se da una serie de líneas guía que permitan mejorar dicho proceso de planificación.

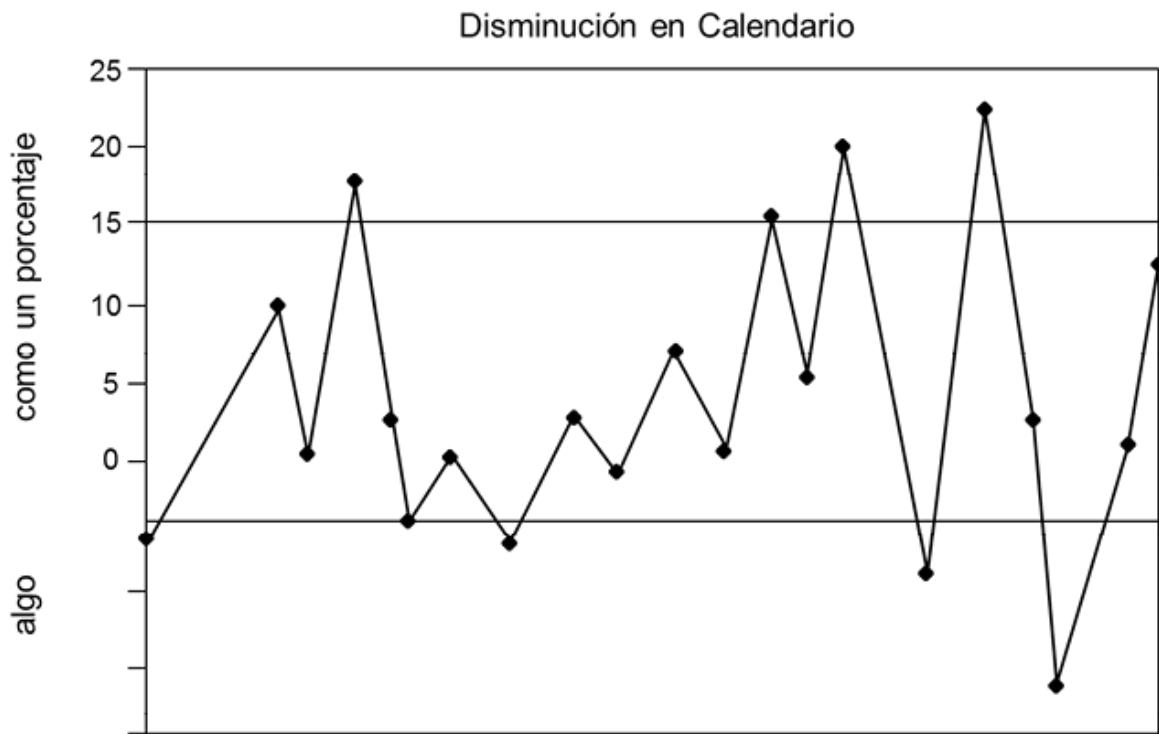


Figura 5.10 Ejemplo SPC evaluación disminución de calendario (Jalote, 1999)

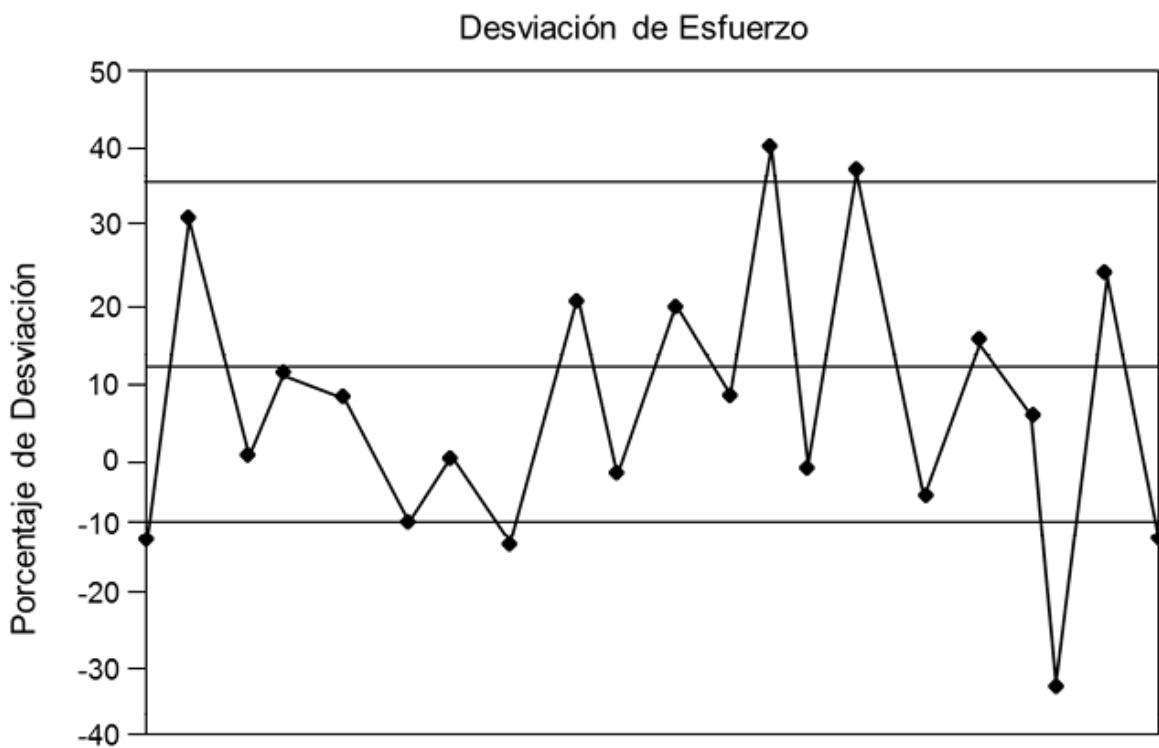


Figura 5.11 Ejemplo SPC evaluación desviación esfuerzo (Jalote, 1999)

Chang y Tong (2013) proponen el uso de gráficos de control Q (Quesenberry, 1991), como un tipo de gráfico más apropiado que los

tradicionales gráficos X y R, para su aplicación en procesos de desarrollo software ante la dificultad de disponer de grandes cantidades de datos. Para ilustrar su uso, lo aplican entre otros procesos al de seguimiento de costes en proyectos software mediante la monitorización de los indicadores SPI (*Schedule Performance Index*) y CPI (*Cost Performance Index*) de la técnica de valor ganado (*Earned Value Technique*) (ver Figura 5.12). En concreto, se usan los datos de proyectos del ITRI (Industrial Technology Research Institute) en Taiwán. Los datos de cada muestra representada en el gráfico de control son recopilados en un intervalo de 15 días. Como resultado, los autores establecen que a diferencia se pueden representar simultáneamente en el mismo gráfico Q, lo que facilita a los gestores del proyecto su evaluación conjunta. Se detecta que el CPI se mantiene bajo control, pero se detecta una anomalía en el indicador SPI, por lo que se sugiere realizar análisis causal para detectar la posible causa de dicha anomalía.

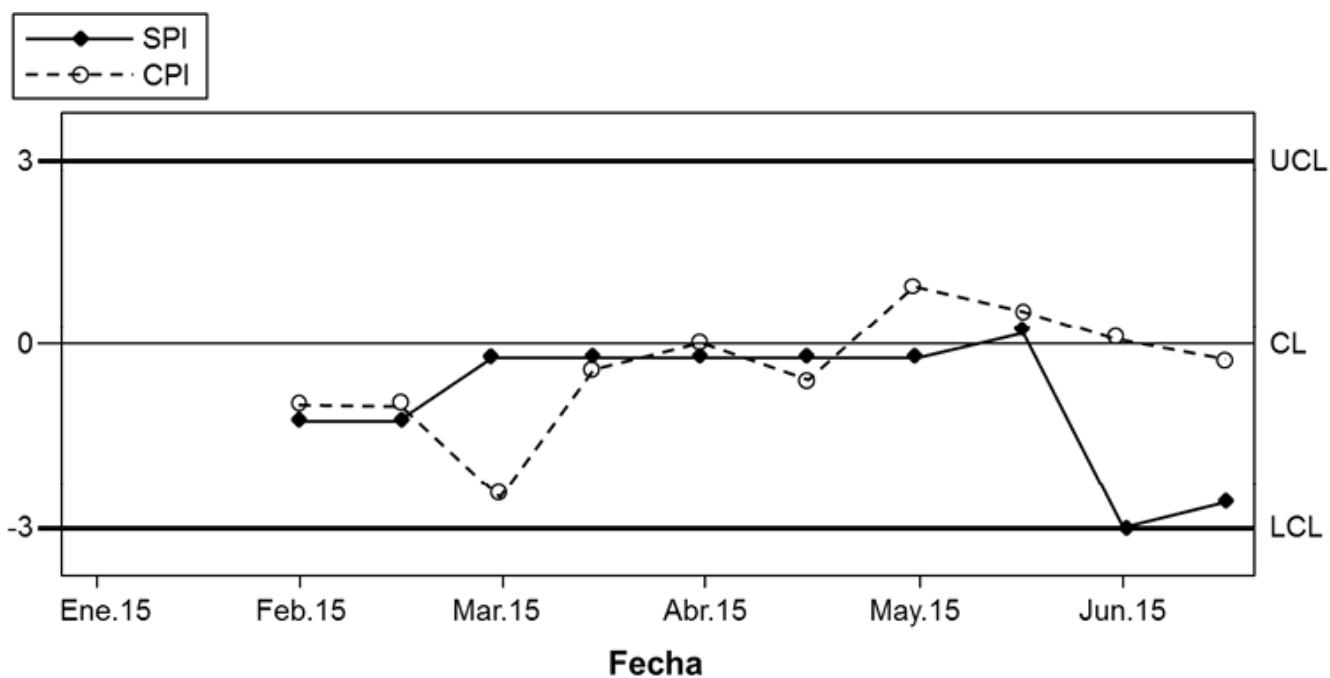


Figura 5.12 Evaluación de indicadores de valor ganado (Chang y Tong, 2013)

Otros trabajos se centran en el proceso de estimación de la fiabilidad software, en concreto Prasad et al. (2011) analizan la idoneidad del método de estimación MMLE (*Modified Maximum Likelihood Estimation*). Se usó un gráfico de medias X, para identificar la variación del proceso de estimación situando en el eje Y la diferencia sucesiva de valores de media acumulativos y en el eje X el número total de fallos (Figura 5.13). Como resultado se identifican los puntos 10 y 25 por debajo del límite inferior de control, lo que indica la necesidad de análisis causal que no es realizado en el contexto de dicho trabajo.

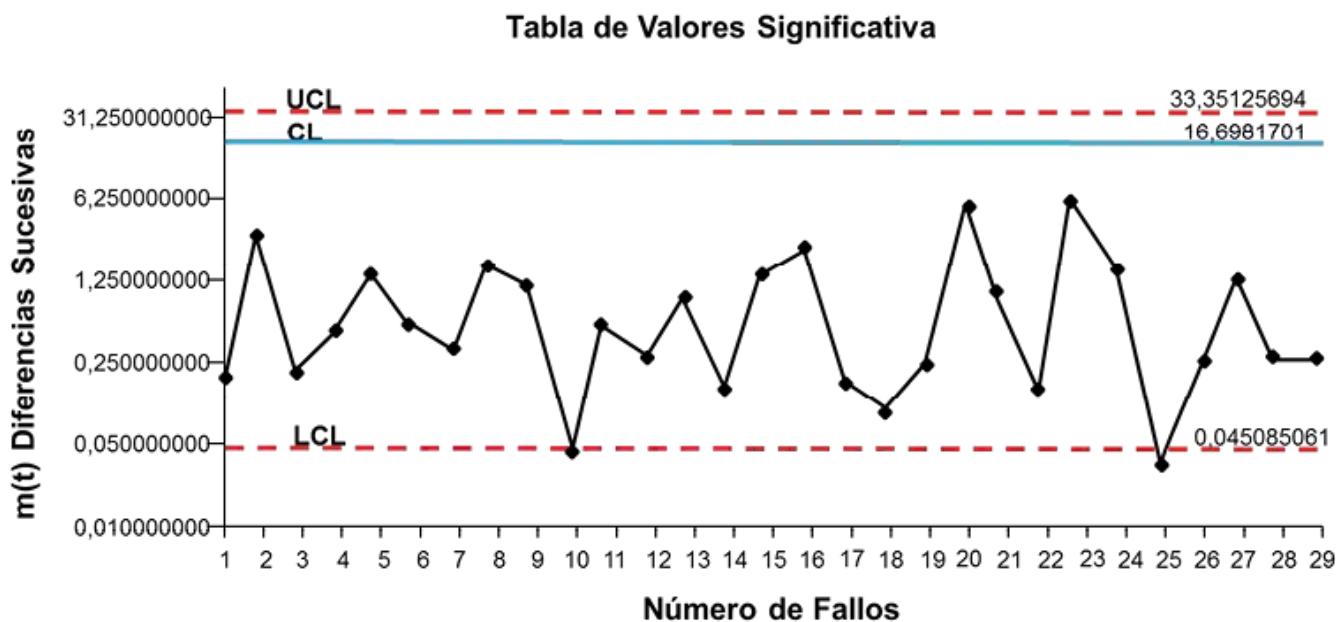


Figura 5.13 Evaluación de fiabilidad (Prasad et al., 2011)

Matias et al. (2016) evalúan la validez externa del método de análisis diferencial de software, que permite detectar el envejecimiento de aplicaciones software en base a su consumo de memoria. Para ello se evaluaron tres versiones de la aplicación de servidor “Squid”, obteniendo que el uso de la pila es un mejor indicador de envejecimiento que otras medidas como el tamaño del conjunto residente (*Resident Size Set, SS*). Para ello se usaron distintas técnicas de procesamiento de datos, concluyendo que SPC obtuvo mejores

resultados mediante la aplicación de gráficos de control de suma acumulada (*Cumulative sum control chart*, CS) y promedio móvil ponderado exponencialmente (*exponentially weighted moving average*, EW).

SPC: Proceso de Inspecciones Software

Paulk (2001) aplica SPC, en particular los gráficos de control XmR, a datos para la evaluación de la aplicación del Proceso Software Personal (PSP). Para la evaluación aplican las siguientes reglas de detección de anomalías (Wheeler y Chambers, 1992):

- **Regla 1:** un punto simple fuera de los límites de control (3 sigma).
- **Regla 2:** al menos dos de tres puntos sucesivos caen al mismo lado y están a más de dos sigmas de distancia respecto a la línea central.
- **Regla 3:** al menos cuatro puntos de cinco caen al mismo lado y más de un sigma de distancia respecto a la línea central.
- **Regla 4:** al menos ocho puntos sucesivos caen al mismo lado de la línea central.

Para ilustrar de forma introductoria la aplicación de SPC en PSP, Paulk (2001) se centra en el proceso de inspecciones, para lo que se usaron varios conjuntos de datos, de los cuales se muestra el análisis de 71 programas en C realizados por estudiantes. Como medidas se usan: el ratio de revisiones, calculado como la división entre número de líneas de código y esfuerzo (en minutos); y densidad de defectos, calculada como el número de defectos detectados dividido por la mediana de tamaño del programa. Como resultado del análisis de ratio de

revisiones, se identificaron siete señales de alarma correspondientes a los puntos de datos 25, 43, 63, 69 y 88, tras cuyo análisis se detectó que se debía al uso de un proceso de inspección inestable debido a la inexperiencia de los estudiantes, que aplicaban por primera vez el proceso de inspecciones según PSP. Estos puntos fueron descartados para el cálculo de los límites de control del gráfico y de control de la densidad de defectos, pero sí fueron usados para su visualización en el gráfico, tras lo cual se identificaron dos señales de alarma correspondiente a los puntos 40 y 41. Tras ello se recomendó realizar un análisis causal antes de decidir si estos puntos correspondían a causas asignables que quedó fuera del alcance de trabajo.

Florence (2001) analiza el proceso de inspecciones por pares a lo largo del ciclo de vida del proyecto, para lo cual se centra en las siguientes medidas de datos de proyectos:

- **Muestra:** serie de revisión por pares
- **Unidades:** número de unidades software revisadas.
- **SLOC:** número de líneas de código fuente revisadas.
- **Defectos:** número de defectos detectados en cada muestra.
- **Defectos / 1000 SLOC:** defectos normalizados a 1000 líneas de código por muestra.

En particular, se ilustra con los siguientes ejemplos la obtención y análisis de gráficos de control:

- **Ejemplo 1:** se analizan 6 muestras (de febrero a abril de 1997), detectándose una alarma (punto fuera de los límites) en el tercer punto de datos, véase Figura 5.14. Mediante la aplicación de análisis causal se determinó que, cuando el proyecto implicaba el

seguimiento de ciertos estándares de codificación, se detectaban numerosas violaciones a la norma debido a la falta de conocimiento sobre estos estándares. Como mejora se propuso aumentar la formación siempre que se introdujera un nuevo proceso o tecnología en el desarrollo de proyectos.

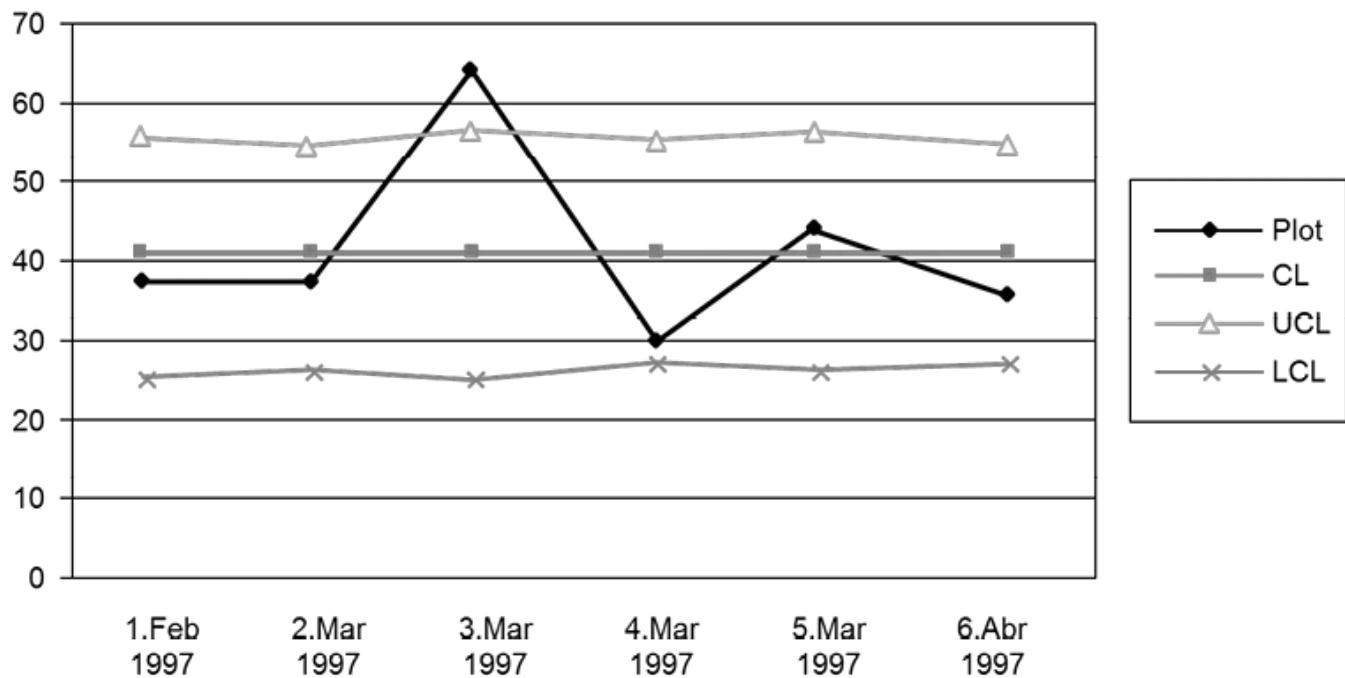


Figura 5.14 Ejemplo de gráfico de control del proceso de inspecciones (Florence, 2001)

- **Ejemplo 2:** se analizan 8 muestras (de marzo a mayo de 1998), detectándose una alarma en el quinto punto de datos. Mediante la aplicación de análisis causal, se determinó que los datos de esa muestra provenían de código de la base de datos mientras que el resto de muestras era código de aplicación. Dado que se debe tener homogeneidad de los datos en los gráficos de control, se eliminó dicha muestra y se volvió a realizar el análisis resultando un proceso bajo control estadístico.

Fernandez-Corrales et al. (2013) también aplican gráficos de control SPC para el proceso de inspecciones analizando como medidas las siguientes:

- Porcentaje de defectos rechazados: porcentaje de defectos clasificados como “rechazados” respecto al total de defectos detectados. Los defectos de este tipo no se consideran por tanto defectos reales.
- Porcentaje de defectos encontrados en operación, que son los detectados por el cliente final respecto al total de los defectos detectados.
- Porcentaje de defectos de criticidad alta detectados en producción, porcentaje de defectos encontrados por el cliente final respecto al total de los defectos detectados.
- Porcentaje de defectos de criticidad alta detectados en pruebas: porcentaje de defectos encontrados por el cliente en la fase de pruebas respecto al total de los defectos detectados. Porcentaje de defectos causados por lógica defectuosa.

Para el análisis aplicaron las ocho reglas de Nelson (1999) para identificar alarmas en los gráficos de control. En particular se aplican los gráficos de control XmR y EWMA (*Exponentially Weighted Moving Average*), proporcionando una comparación entre ambos en los que se deduce que debido a los límites más cercanos obtenidos con gráficos EWMA hay más sensibilidad a pequeños incrementos en el porcentaje de defectos en comparación con los gráficos XmR.

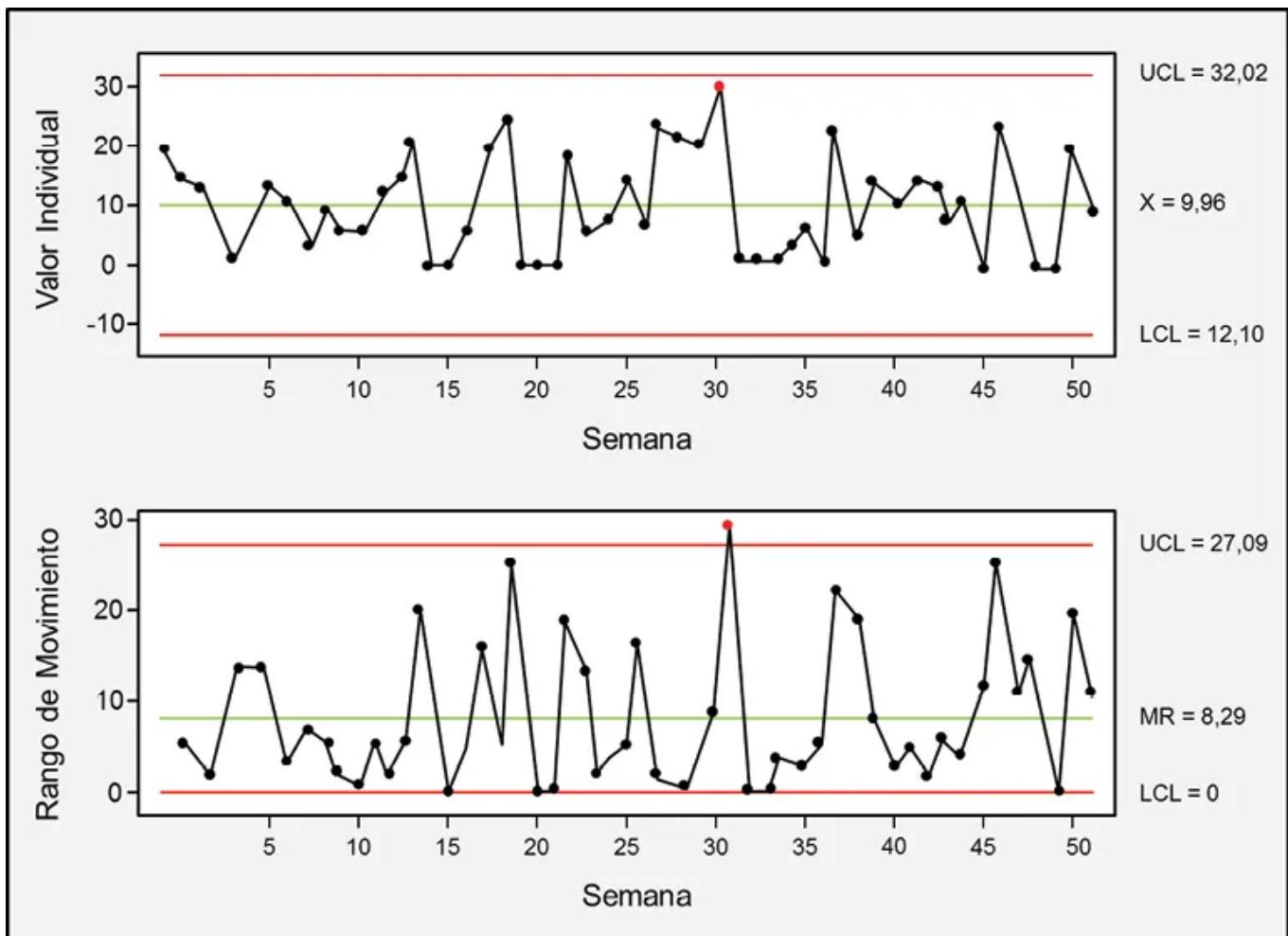


Figura 5.15 Ejemplo de gráfico de control XmR para analizar el porcentaje de defectos rechazados (Fernandez-Corrales et al., 2013)

SPC: Proceso de Pruebas

El proceso de pruebas es otro de los procesos relevantes sobre los que podemos encontrar diversas aplicaciones de SPC. A continuación, se describen algunos casos a modo de ejemplo.

Weller (2000), además de aplicar SPC para la evaluación del proceso de inspección, ilustra la aplicación de SPC en la etapa de pruebas. En particular se aplica SPC para las pruebas de sistema con el fin de responder la pregunta de cuándo se finalizarán las pruebas. Para ello se puede usar la estimación de defectos que permanecen en el producto y

el ratio de eliminación de defectos. Se utilizaron como gráficos de control los gráficos XmR y se evaluaron dos proyectos. En la Figura 5.16 se muestra el gráfico de control del ratio de llegada de problemas (defectos potenciales). El límite inferior se fija a cero al ser un valor negativo. En este caso de estudio, tal como reconoce el propio autor, es necesario disponer de muestras mayores para desarrollar gráficos de control, normalmente de 16 a 20, pero no siempre es posible encontrarlas en la práctica. Con menos de 15 puntos de datos hay más probabilidad de que puntos que están cerca de los límites den señales incorrectas de proceso fuera de control.

Tal como se puede observar en la Figura 5.16, el ratio de llegadas está por debajo del límite superior hasta la semana 10 para ambos proyectos. En la semana 11 se puede observar que el ratio de problemas sube a 33, lo que supera el límite superior y sugiere buscar la causa especial. En este caso, el resultado se debió a que la fase de pruebas de sistema comenzó en la semana 11 y usó un conjunto de casos de prueba más robusto y configuraciones de prueba mayores. Por ello, dado que una nueva fase de pruebas había comenzado, se recalcularon los límites de control a partir de esa semana, tal como muestra la Figura 18.10.

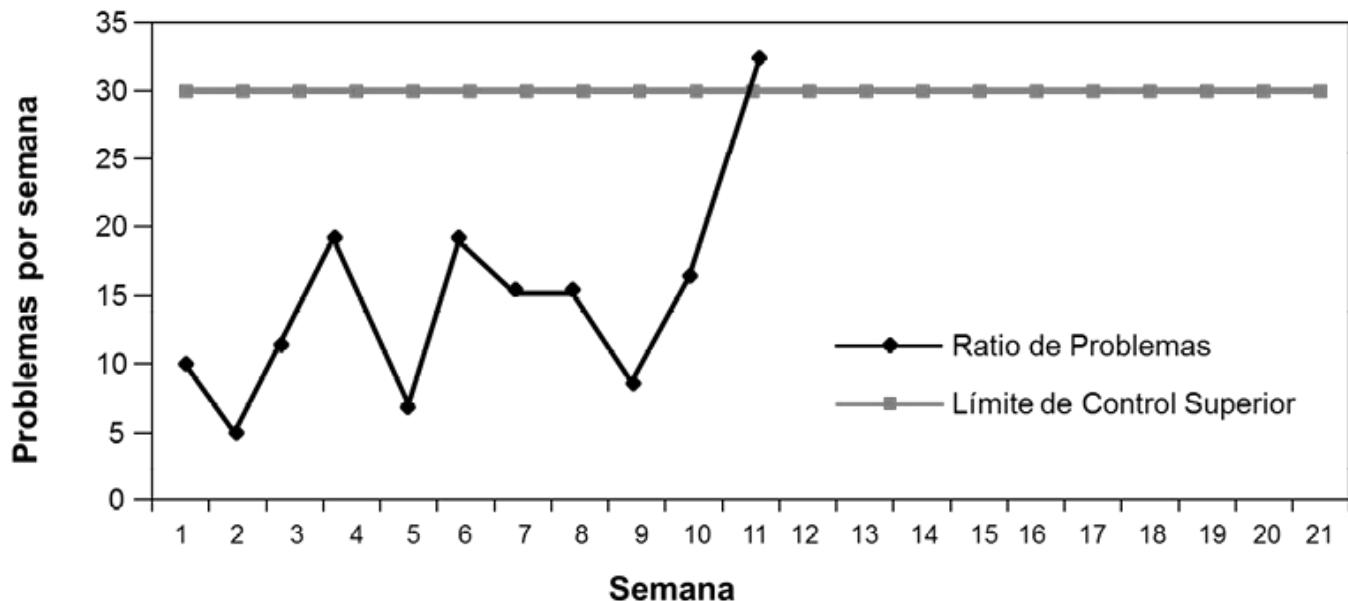


Figura 5.16 Ejemplo de gráfico de control para analizar la ratio de llegada de problemas (Weller, 2000)

En base a los datos mostrados en la gráfica de la Figura 5.17, se deduce que con SPC no se puede responder a la pregunta de cuándo finalizará la etapa de pruebas, que podría malinterpretarse como cercano a la semana 20 debido a la tendencia a la baja de los valores. Pero con SPC, lo que se determina es si el proceso está bajo control o no, y tal como se muestra en la figura lo está si sus valores de problemas por semana oscilan entre 2,4 (LCL) y 27 (UCL). Si se observa la semana 24, el valor cae por debajo del límite inferior, lo que indica una causa assignable de variación. Esto podría ser el indicador de fin de pruebas si el análisis causal determina que la causa está relacionada con el producto (y no, por ejemplo, con una semana más corta de trabajo o falta de progreso del trabajo debido a situaciones de bloqueo, etc.).

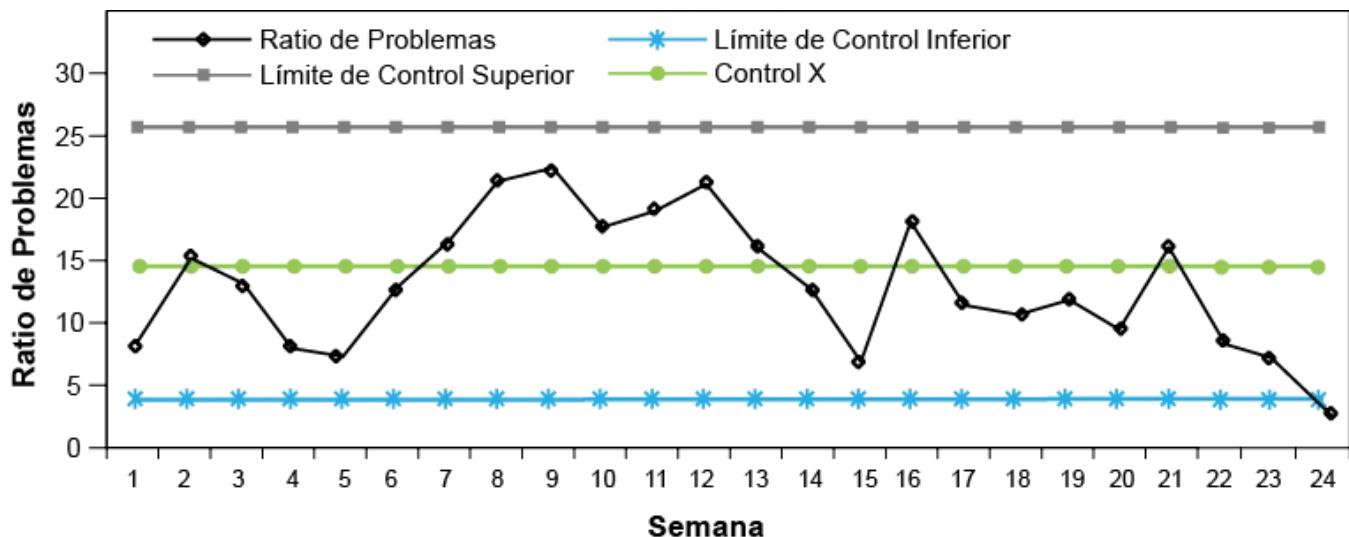


Figura 5.17 Ejemplo 2 de gráfico de control para analizar el ratio de problemas (Weller, 2000)

Como conclusión, Weller (2000) establece que SPC es una técnica que puede ser útil en procesos software, que los cálculos son relativamente sencillos y en su experiencia en los dos proyectos reportados en este ejemplo a partir del análisis SPC se consiguió reducir la densidad de defectos de las nuevas versiones del producto 10 veces más que en versiones anteriores.

Card y Berg (1989) desarrollan gráficos de control X para analizar medidas sobre defectos encontrados en la fase de pruebas en dos grandes proyectos de la NASA (promedio de 850.000 líneas de código). En particular se evalúa la tasa de errores totales (errores por cada mil líneas de código entregadas) informados en la fase de pruebas de sistema, de aceptación y de operación. En sus resultados detectaron anomalías en la versión 7 del producto de uno de los proyectos mientras que el otro se encontraba bajo control estadístico.

Jalote (1999) evalúa el proceso de pruebas software aplicando SPC, para lo cual ilustra con un gráfico de control X la densidad de defectos

obtenida en el proceso de pruebas sobre programas desarrollados en Visual Basic.

Komuro (2006) aplica SPC en una empresa de desarrollo de software de alto nivel en Japón. En concreto, evalúa el proceso de revisión por pares y se incluyó dentro del control el proceso de prueba donde la medida a controlar fue el índice de errores, además de analizar la densidad de defectos en la fase de pruebas del producto final. Para el proceso de revisión por pares, las medidas a controlar fueron la velocidad de revisión, la densidad de defectos, detección temprana de errores (porcentaje del número de defectos detectados por revisión en pares con respecto al número total de defectos encontrados durante todo el proceso de desarrollo) y la eficiencia de revisión (número de errores detectados en la revisión por pares por persona-hora dedicada en la revisión).

Yamada (2016) propone un gráfico de control para la aplicación de SPC en entornos de desarrollo de proyectos *open source*, para lo cual se analizan los defectos software encontrados por el sistema de detección y seguimiento de defectos (*bug tracking system*) en el desarrollo de Android 1.5. A partir de los resultados obtenidos proponen una forma de estimar el tiempo óptimo en el que parar el desarrollo de un proyecto *open source* y transferirlo a operaciones.

LECTURAS RECOMENDADAS

- *Florac, W. A. y Carleton, A. D. (1999). Measuring the Software Process. Statistical Process Control for Software Process Improvement. Addison Wesley.*

En este libro se presentan los fundamentos de SPC en Procesos Software y se incluyen diversos ejemplos de aplicación.



Dogram Code

Accede a la Biblioteca Online +300 libros en PDF Gratis!

<https://dogramcode.com/biblioteca>

Únete al Canal de Telegram

https://t.me/bibliotecagratis_dogramcode

Únete al grupo de Facebook

<https://www.facebook.com/groups/librosyrecursosdeprogramacion>

<https://dogramcode.com/programacion>

6

IMPLANTACIÓN DE LA MEDICIÓN

INTRODUCCIÓN

En este capítulo abordaremos algunos desafíos a los que se enfrentan las organizaciones que quieren implantar la medición. Para ello utilizaremos los conocimientos obtenidos de varios proyectos con organizaciones, tanto grandes como pequeñas.

La implantación de la medición de procesos, proyectos, productos y servicios software parece sencilla y se ha visto favorecida por la aparición, de decenas de entornos y herramientas de medición. Sin embargo, la medición plantea enormes desafíos a las organizaciones, que terminan en muchas ocasiones abandonando los programas de medición o no obteniendo ninguna ventaja con su implantación.

Empezaremos por repasar cómo plantean la medición los principales estándares internacionales para posteriormente abordar los desafíos, problemas, factores críticos de éxito y finalizar proponiendo un método para la implantación de la medición.

MEDICIÓN EN LOS ESTÁNDARES INTERNACIONALES

ISO/IEC/IEEE 12207

En la norma ISO/IEC 12207 (ISO, 2017a), las actividades que se pueden realizar durante el ciclo de vida del software se agrupan en cuatro grandes categorías: procesos de acuerdo, procesos organizacionales de proyecto, procesos de gestión técnica, y procesos técnicos. Dentro de los procesos de gestión técnica se encuentra el proceso de medición, que es el encargado de “recoger, analizar y reportar datos e información objetivos para dar soporte a una gestión efectiva y demostrar la calidad de productos, servicios y procesos”.

Este estándar señala como resultados del proceso de medición que: 1) se identifican las necesidades de información, 2) se identifican o desarrollan un conjunto apropiado de medidas basadas en las necesidades de información, 3) se recogen, verifican y almacenan los datos requeridos, 4) se analizan los datos y se interpretan los resultados, y 5) los elementos de información proporcionan información objetiva que soporta las decisiones.

Además, ISO/IEC 12007 propone dos actividades para este proceso con las siguientes tareas:

- 1. Preparar la medición:** definir la estrategia de la medición, describir las características de la organización que son relevantes para la medición (como los objetivos de negocio y los técnicos), identificar y priorizar las necesidades de información, seleccionar y especificar medidas que satisfagan las necesidades de información, definir los procedimientos de recolección, análisis, acceso y reporte de los datos, definir los criterios para evaluar los elementos de información y el proceso de medición, e identificar y planificar el uso de los sistemas o servicios habilitadores necesarios.
- 2. Llevar a cabo la medición:** integrar los procedimientos manuales o automatizados para la generación, recolección, análisis y reporte de datos en los procesos relevantes: recoger, almacenar y verificar los datos;

analizar los datos y desarrollar los elementos de información; y registrar los resultados e informar a los usuarios de la medición.

Medición en CMMI V2.0

En la actualidad, el CMMI (*Capability Maturity Model Integration*) se considera “un conjunto integrado de buenas prácticas que permite a las organizaciones mejorar el desempeño de sus procesos de negocio clave, proporcionando una hoja de ruta clara para construir, mejorar y mantener la capacidad” (CMMI, 2018). Además, proporciona un conjunto de “vistas” con los siguientes modelos predefinidos (aunque se pueden elaborar modelos nuevos que combinen parte de los anteriores):

- *CMMI Development* (CMMI-DEV), CMMI para Desarrollo, con el fin de mejorar los procesos y el desempeño desarrollando mejores productos y servicios.
- *CMMI Services* (CMMI-SVC), CMMI para Servicios, con el fin de mejorar las capacidades y procesos para proporcionar un mejor desempeño de los servicios.
- *CMMI Supplier Management* (CMMI-SPM), CMMI para la Gestión de Proveedores, con el fin de mejorar los procesos y el desempeño para optimizar la cadena de suministro.
- *CMMI People Management* (CMMI-PPL), CMMI para la Gestión de Personas, con el fin de mejorar los procesos y el desempeño para gestionar personas.

Cada uno de los modelos CMMI está compuesto por áreas de prácticas, algunas comunes y otras específicas para cada uno. Según CMMI (2018), un “área de prácticas”, es “un conjunto de prácticas similares que juntas consiguen un objetivo definido, valor y la información requerida”. Una práctica en CMMI V2 consta de dos partes: la información requerida (para entender el propósito y el valor de la práctica, lo que incluye el enunciado de la práctica, el enunciado de valor y la información adicional requerida) y la información explicativa (para entender mejor la información requerida y que incluye actividades y productos de trabajo de ejemplo).

Las prácticas se agrupan según diferentes criterios; así, un “Grupo de Prácticas”, es “una estructura organizativa para las prácticas de un área con el fin de facilitar su comprensión y adopción y proporcionar una ruta para la mejora del desempeño”. Los grupos de prácticas que se contemplan son niveles evolutivos que proporcionan un camino de mejora de desempeño que se caracterizan por (CMMI, 2018):

- Nivel 0: Enfoque incompleto para abordar el propósito del área de prácticas, y se puede o no satisfacer el propósito de todas las prácticas.
- Nivel 1: Prácticas básicas que describen un enfoque inicial para abordar el propósito del área de prácticas, no existe un conjunto completo de prácticas para satisfacer el propósito completo del área, lo que se espera de una organización o proyecto que empieza su camino hacia la mejora, y que empieza a centrarse en cuestiones de desempeño.
- Nivel 2: Conjunto simple, pero no completo, de prácticas para satisfacer el propósito del área, no se requiere el uso de activos o estándares organizacionales, el propósito del conjunto de prácticas puede variar de maneras diferentes dependiendo del proyecto, y se identifica y monitoriza objetivos de desempeño de proyecto.
- Nivel 3: Utiliza estándares organizacionales e incluye la adaptación (*tailoring*) de procesos para abordar las características únicas de proyecto y trabajo, se utilizan y se contribuye a los activos organizacionales, y se gestiona tanto el desempeño de proyecto como el organizacional.
- Nivel 4: Utilización de la estadística y otras técnicas cuantitativas para detectar o refinar el área de interés o predecir si se lograrán los objetivos de calidad y desempeño del proceso, y se comprende la variación del

desempeño estadística o cuantitativamente y se gestiona el progreso hacia los objetivos de calidad y desempeño.

- Nivel 5: Utilización de la estadística y otras técnicas cuantitativas para optimizar el desempeño y mejorar el logro de los objetivos incluyendo la medición y desempeño de la organización, y calidad y desempeño de los procesos.

Un “Área de Capacidad” es “un grupo de áreas de prácticas relacionadas que pueden proporcionar un desempeño mejorado en las habilidades y actividades de una organización o proyecto”. Una “Categoría” es un grupo lógico de vistas de áreas de capacidad relacionadas que abordan problemas comunes encontrados por la organización al producir o entregar soluciones.

Dentro de la categoría “mejorar” (*improving*), se encuentra el área de capacidad IMP – Mejora del desempeño, que engloba tres áreas de prácticas: Gestión de Procesos (PCM), Desarrollo de Activos de Proceso (PAD), y **Gestión del Desempeño y la Medición (MPM)**.

El propósito del área MPM (*Managing Performance and Measurement*) de Gestión del Desempeño y la Medición, es “gestionar el desempeño utilizando la medición y el análisis para lograr objetivos de negocio, lo que lleva consigo fijar objetivos para el negocio, la medición y el desempeño, y el desempeño de los procesos y la calidad”.

En la tabla 6.1 se resumen las prácticas de Gestión del Desempeño y la Medición por nivel de capacidad según CMMI (2018).

Nivel	Cod. Prac.	Práctica
1	MPM 1.1	Recoger medidas y registrar desempeño
	MPM 1.2	Identificar y abordar cuestiones de desempeño
2	MPM 2.1	Derivar y registrar objetivos de medición y desempeño a partir de necesidades y objetivos de negocio seleccionados y mantenerlos actualizados
	MPM 2.2	Desarrollar, mantener actualizados y utilizar definiciones operacionales para las medidas
	MPM 2.3	Obtener datos de medición especificados según las definiciones operacionales
	MPM 2.4	Analizar los datos de desempeño y medición de acuerdo con las definiciones operacionales
	MPM 2.5	Almacenar datos de medición, especificaciones de medición y resultados de análisis de acuerdo con las definiciones operacionales
	MPM 2.6	Tomar acciones para abordar las cuestiones identificadas para cumplir los objetivos de medición y desempeño
3	MPM 3.1	Desarrollar, registrar, mantener actualizados, y utilizar sistemas y acuerdos de servicio estándares en la organización
	MPM 3.2	Seguir procesos y estándares organizacionales para desarrollar y utilizar definiciones operacionales para las medidas y mantenerlas actualizadas
	MPM 3.3	Desarrollar, mantener actualizado y seguir un proceso de calidad de datos
	MPM 3.4	Desarrollar, mantener actualizado y utilizar un repositorio de medición de la organización
	MPM 3.5	Analizar el desempeño organizacional utilizando datos de medición y desempeño para determinar las necesidades de mejora del desempeño
	MPM 3.6	Comunicar periódicamente los resultados de desempeño a la organización

	MPM 4.1	Utilizar técnicas estadísticas u otras técnicas cuantitativas para desarrollar, mantener actualizados y comunicar los objetivos de desempeño de procesos y calidad que son trazables a objetivos de negocio
4	MPM 4.2	Seleccionar las medidas y técnicas analíticas para gestionar cuantitativamente el desempeño para conseguir los objetivos de desempeño de los procesos y la calidad
	MPM 4.3	Utilizar técnicas estadísticas u otras técnicas cuantitativas para desarrollar y analizar líneas de referencia (<i>baselines</i>) de desempeño de procesos y mantenerlas actualizadas
	MPM 4.4	Utilizar técnicas estadísticas u otras técnicas cuantitativas para desarrollar y analizar modelos de desempeño de procesos y mantenerlos actualizados
	MPM 4.5	Utilizar técnicas estadísticas u otras técnicas cuantitativas para determinar o predecir el logro de los objetivos de desempeño de procesos y calidad
	MPM 5.1	Utilizar técnicas estadísticas u otras técnicas cuantitativas para asegurar que los objetivos de negocio se alinean con la estrategia y desempeño de la organización
5	MPM 5.2	Analizar datos de desempeño utilizando técnicas estadísticas u otras técnicas cuantitativas para determinar la capacidad de la organización para satisfacer los objetivos de negocio seleccionados e identificar áreas potenciales de mejora del desempeño
	MPM 5.3	Seleccionar e implementar propuestas de mejora, basadas en el análisis estadístico y cuantitativo sobre los efectos esperados de las mejoras propuestas para cumplir los objetivos de negocio, calidad y desempeño de procesos

Tabla 6.1. Prácticas de MPM por nivel de capacidad según CMMI (2018).

PRINCIPALES DESAFÍOS

El objetivo de implantar la medición suele ser mejorar algún aspecto relacionado con la calidad o productividad de algún proceso del ciclo de vida del software. Un programa de medición tendrá éxito si se obtienen beneficios efectivos respecto a un problema, a partir de la información proporcionada por el programa de medición.

Sin embargo, demostrar este beneficio respecto al coste de implementación no es tarea fácil y el hecho de no detectar con claridad los beneficios del programa de medición respecto a su coste provoca desconfianza en su implantación (Solingen y Berghout, 2001). Para vencer esta resistencia, estos autores sugieren medir el esfuerzo de implementación del programa de medición, los beneficios eficientes y otros ingresos adicionales y expresarlos en valores económicos.

Ebert et al. (2004) resaltan los siguientes riesgos a los que están sometidos los programas de medición:

- **Recopilar medidas sin significado.** La medición debe estar siempre orientada a los objetivos, es decir, debe ser “*goal driven*”. Por lo tanto, cada medida debe tener una aplicación práctica, ya que en caso contrario solo supondría malgastar esfuerzo para obtenerla y puede provocar pérdida de motivación en los programas de medición.
- **No analizar las medidas.** Medir se puede considerar relativamente “sencillo”, pero trabajar con números supone un esfuerzo poco reconocido. Por lo general, las medidas se incluyen en informes con gráficos y tablas, pero esto es insuficiente, ya que es necesario realizar análisis más profundos que permitan relacionar las medidas con los objetivos y el desempeño.
- **Establecer objetivos poco realistas.** Muchos gestores pueden estar muy entusiasmados con programas de medición y establecer un alineamiento entre los números y los objetivos, pero lo importante es que los objetivos sean realistas, se basen en la experiencia y situación de la organización y no conduzcan a una organización a “estar continuamente al límite”.
- **Parálisis por análisis.** Situación que puede ocurrir cuando las medidas se usan de forma independiente de los ciclos de vida y procesos de software. Se recopila mucha información de forma frecuente y se malgasta mucho tiempo en burocracia. La clave está en entender que la medición no es una actividad separada sino una parte clave de los procesos de ingeniería y gestión.

En cuanto a los obstáculos con los que nos encontramos a la hora de implementar programas de medición, que se agravan especialmente en la pequeña y mediana empresa (Gresse et al., 2003; Díaz-Ley et al., 2008; Tahir et al., 2018), destacamos los siguientes:

- **Limitación de recursos y tiempo:** cuando el tiempo y los recursos de los proyectos son ajustados, no se dedica esfuerzo a las tareas de medición y suele ser la primera actividad que se deja sin hacer (Komi-Sirviö et al., 2001).
- **Dudosa fiabilidad de las actividades de medición:** en muchos casos, los procesos de recolección de datos, análisis, interpretación y comunicación de resultados son insuficientes, mal organizados y poco fiables lo que provoca desconfianza en el uso del programa de medición (Selby, 2005).
- **Escasa formación y conocimiento en medición:** si hay limitaciones de presupuesto no se suele formar al personal en el área de medición del software. Esto hace que no haya cultura en la empresa respecto a la medición del software, que las personas desconozcan sus beneficios y relacionen la medición con la idea de controlar al personal y, por lo tanto, desconfíen del uso de tales iniciativas.
- **Poco conocimiento en el área de medición del software:** normalmente la cultura de la medición es pobre. Un programa de medición efectivo necesita mucho tratamiento de información, modelos, toma de decisiones, etc., por lo tanto, se convierte en una tarea complicada y más aún cuando las personas que lo llevan a cabo no tienen una formación extensa en esta área (Briand et al., 1996a).
- **Coste:** una de las principales causas que impiden el uso de los programas de medición es que los beneficios que se obtienen de éste no son claros respecto a su coste (Berry et al., 2004).

FACTORES CRÍTICOS DE ÉXITO

En este apartado mostramos los factores generales y más destacados que influyen en el éxito de un programa de medición. Estos factores de éxito se clasifican en dos tipos: técnicos y organizativos.

Los **factores técnicos** identificados son los siguientes:

- Disponer de un sistema para la captura de métricas, que sea sistemático y orientado al proceso, que haga que los jefes de proyecto tengan una percepción de seriedad y fiabilidad respecto a la actividad de medición y, por lo tanto, promuevan su uso.
- Emplear herramientas automatizadas para facilitar el uso de las métricas.
- Optimizar los procesos de captura para mejorar la precisión de los datos obtenidos, de forma que no disminuya la productividad del desarrollador debido a esta actividad.
- Utilizar métricas precisas y relevantes en el contexto.
- Emplear procedimientos de análisis formales y útiles para proporcionar más información y más precisa para la toma de decisiones.
- Formar en la medición del software, ya que las personas que conocen los beneficios de la medición usarán más estas técnicas que aquellos que no están formados en estos aspectos.
- Comunicar de manera adecuada y precisa la información obtenida de la medición.

Por su parte, los **factores organizativos** identificados son los siguientes:

- Acuerdo del personal con los objetivos de medición.
- Asignación de los recursos suficientes para el desarrollo de las actividades de medición.
- Apoyo de la dirección, para que se establezcan y perduren las iniciativas de medición.

- Suficiente nivel de madurez de la organización. Las empresas muy maduras tienen los procesos de medición mejor establecidos, harán mejor uso de la medición para la toma de decisiones y por lo tanto obtendrán mayor beneficio en el desempeño de sus procesos.
- Cultura de la compañía: el modo de trabajar establecido en la organización es un factor determinante en el uso de cualquier técnica y por lo tanto en el uso de las actividades de medición.

Daskalantonakis et al. (1990) en el contexto de su propuesta de método de madurez para evaluar la tecnología de la medición del software, destacan también otro factor determinante para el éxito de la implantación de un programa de medición en una organización: su integración con el resto de procesos de software ya establecidos. Además, expone que los jefes de proyecto e ingenieros involucrados en el proyecto son las personas más adecuadas para analizar los datos recogidos en el programa de medición, ya que son los expertos en el dominio y pueden interpretar perfectamente el significado de los datos.

Por otro lado, Hall y Fenton (1997) identifican quince factores de éxito para la implantación de los programas de medición de los cuales ocho han sido expuestos, mientras que los siete restantes son los siguientes:

- Implementar el programa de medición gradualmente: es muy complicado por no decir imposible abordar gran cantidad de objetivos de medición de una vez, por lo que se aconseja implantar poco a poco las actividades de medición en la organización y, mientras, se van mejorando. Otros autores (Wieggers, 1999; Bassman et al., 1995) recomiendan comenzar los programas de medición desarrollando objetivos comunes y conocidos además de necesarios en la organización.
- Disponer de un marco de medición bien planificado donde se asegure el adecuado flujo de información. Respecto a este punto, la mayoría de los autores (DoD, 2000; Niessink y Vliet, 2001) defienden la idea de definir los programas de medición orientados a objetivos.
- Reutilizar, en la medida de lo posible, el material de medición que se tenga ya disponible.
- Involucrar a los desarrolladores durante la implementación.
- Hacer que el proceso de medición sea transparente para los desarrolladores.
- Mejorar el programa de medición constantemente.
- Contratar a “gurús” externos especialistas en la medición del software.

El último factor de éxito que destacaremos es que hay que asegurar que los programas de medición no se orienten a evaluar a las personas ya que en tal caso seguramente terminará perdiendo su validez. El personal se desconcentrará de su trabajo y registrará datos positivos y en algún caso devaluados (Solingen et al., 1997).

Tahir et al. (2016) destaca en la revisión sistemática de decenas de programas de medición, la importancia de la sostenibilidad del programa de medición y de que la medición esté orientada a objetivos, de forma que exista una sincronización entre los objetivos de medición y los objetivos operacionales, organizacionales, de negocio y de los stakeholders.

Tras los factores analizados en este apartado podemos concluir que, aunque actualmente no se dispone de unas leyes universales que nos permitan saber con precisión qué técnicas aplicar para implantar exitosamente los programas de medición (Briand et al., 2002), es claramente positivo y recomendable evaluar el estado de la organización respecto a la aplicación de programas de medición con el fin de detectar imposibilidades, deficiencias y actuar consecuentemente. Esta evaluación se puede realizar revisando los factores de éxito indicados anteriormente.

CONSEJOS PRÁCTICOS

Como indica Pandian (2003), el verdadero significado de la medición se percibe cuando surge dentro de la cultura de la organización; por lo que la aplicación de los métodos, técnicas, aproximaciones y medidas existentes es análoga a cuando se implanta un modelo de calidad, es decir, se debe adaptar a la cultura de la organización. Staron y Meding (2018) destacan que un programa de medición es un “conjunto socio-técnico de sistemas de medición y sus usuarios”, por lo que hay que considerar tanto el aspecto técnico como el social u organizativo.

En este apartado enumeramos algunos consejos que surgen de los factores críticos de éxito expuestos anteriormente y en base a la experiencia en la industria.

Consejos de gestión

1. Comenzar los programas de medición dando soporte a la mejora de procesos. Los programas de medición comienzan con objetivos de mejora, de modo que los objetivos estratégicos deben dividirse en objetivos de proyecto y estos deben estar alineados con los objetivos de cada departamento y los contenidos de los planes de calidad. Del mismo modo, las mediciones y el control de los proyectos deben estar motivados por objetivos concretos y alcanzables (Ebert et al., 2004). Estos objetivos de mejora deben ser realistas en función de la madurez de la organización, de modo que una vez que el proceso de medición sea estable, la organización puede permitirse desarrollar programas de medición para determinados objetivos de negocio, pero si la organización no es suficientemente madura, sería complicado desarrollarlos y la probabilidad de fracaso sería alta, al igual que su coste.
2. Integrar la medición en los procesos de desarrollo. Como señala Daskalantonakis (1992), los programas de medición que se establezcan en la organización deben estar integrados en los procesos de desarrollo. Por ejemplo, si se definen métricas de evaluación de la calidad del software, al final del proyecto éstas deben mostrarse en los informes de cierre de proyecto y por lo tanto deben contemplarse en la plantilla de cierre de proyecto. Además, el programa de medición debe adaptarse a las limitaciones de los procesos de desarrollo; por ejemplo, si el proceso de desarrollo no contempla establecer de manera formal y con un criterio establecido la fiabilidad de un producto, el programa de medición en principio no debería poder evaluar formalmente la fiabilidad de éste.
3. Comenzar de forma inmediata y con un pequeño alcance (Ebert et al., 2004). La puesta en marcha de un programa de medición supone mucha comunicación y formación, de modo que se sugiere realizar sesiones de arranque (“kick off”) para que los equipos técnicos y de gestión entiendan y usen la medición de forma consistente y se debe proporcionar formación sobre todo a gestores senior que evite el uso inadecuado de las medidas en su labor.
4. Contar con el personal adecuado. Los recursos humanos constituyen el factor de éxito clave en todo programa de medición, por lo que entre otros aspectos importantes es necesario dar una adecuada formación tanto a los desarrolladores técnicos como al personal de gestión (Ebert et al., 2004), ya que si las medidas se usan de forma inadecuada el coste del programa de medición será mucho mayor que el coste de formación. Del mismo modo, es recomendable implicar a todas las personas relacionadas con el desarrollo software en el proceso de medición, considerándolo una tarea más a realizar en su trabajo. Así se establece la cultura de medición, se percibe su utilidad, se realiza la toma de decisiones de manera efectiva y se realizan las mejoras en los procesos de software oportunos.
5. Adaptar el programa de medición a la madurez de la organización respecto a la medición del software. Una organización poco madura no puede plantearse, por ejemplo, obtener del programa de medición resultados empíricos de estimación. Esto podría dar lugar a resultados falsos, inesperados, difíciles de repetir en los siguientes estudios y difíciles de interpretar (Briand et al., 2002). Ello implica en empresas poco maduras en

medición de software la necesidad de no involucrar a demasiadas personas en la definición del programa de medición en las primeras fases. La propuesta consistiría en que el grupo de analistas de medición desarrollen un borrador del programa de medición junto con un grupo de personas, de alto cargo, promotores de la iniciativa y que conozca bien los procesos, proyectos y productos del software llevados a cabo en la organización. Una vez realizado el primer borrador, se involucraría a desarrolladores claves y jefes de proyecto que vayan a hacer uso del programa de medición, quienes revisarán el borrador. Tras esta revisión el grupo de analistas actualizan y formalizan el programa de medición. Con esta propuesta es de esperar que el programa de medición se defina de manera más dinámica ya que no todos los interesados están involucrados desde las primeras fases. Además, cuando no se tiene mucha experiencia en la medición del software, la información que se obtiene al principio puede no ser útil debido a la desconfianza en el programa de medición. En Díaz-Ley et al. (2010) se presenta un modelo de madurez para pymes que puede ayudar en este aspecto.

6. Reutilización de los modelos de medición y adaptación a los proyectos específicos. Este principio fue expuesto por Basili y Rombach (1988) y Schneidewind (1999) e indica la conveniencia de definir programas de medición que sean modelos válidos para todos los proyectos o productos que se desarrollan en la empresa. De acuerdo a este principio, definir programas de medición generales facilita el establecimiento del programa de medición en la organización o en la unidad de negocio y, posteriormente, el control de los procesos. Cuando surja un proyecto o producto habría que realizar pequeñas modificaciones para adaptar el programa general de medición al proyecto o producto específico o en caso de que éste tenga alguna necesidad de información particular, se hará un pequeño programa de medición específico, pero no se establecerá en el proceso general de medición de la organización. Un ejemplo sería el de un programa de medición que analiza la fiabilidad de los productos resultante de un proyecto. Si en general este programa se basa en los datos de los fallos del producto en producción (registrados en una herramienta de gestión de incidencias), puede que haya que adaptar el programa de medición al proyecto específico para aquellos productos que no utilicen el mismo gestor de incidencias, o en el caso de que hubiera incidencias asociadas a un determinado producto en las que se asocian atributos específicos a tener en cuenta (la frecuencia en la que se produce el mismo fallo, fecha de ocurrencia, etc.).

Consejos técnicos

1. Uso de métricas existentes y fáciles de recopilar. Es recomendable hacer uso de las medidas proporcionadas por las herramientas de desarrollo, gestión de configuración, gestión de incidencias, gestión de recursos, etc. que pueda haber en la organización y evitar un proceso de recopilación de información complicado que puede llevar a inexactitudes o esfuerzos extra para los desarrolladores y jefes de proyecto. Del mismo modo es importante evitar “trampas estadísticas” (Ebert et al., 2004), considerando que las medidas se caracterizan por una escala y distribuciones estadísticas que determinan su utilidad y modo de aplicación. Por ejemplo, en determinadas situaciones un análisis por cuartiles es más adecuado que por medias, o en otras, junto con la media de un conjunto de valores es deseable mostrar también sus valores máximo y mínimo. Desouza (2015) recomienda evitar métricas “sandía”, que son verdes por fuera (reflejando éxito) y rojas por dentro (fracaso), lo que sucede con métricas que son índices que combinan muchas métricas. Forsgren y Kersten (2018) proponen completar las métricas basadas en sistemas (automáticas) que son precisas, permiten una visibilidad continua y diferentes granularidades, son escalables, etc. con métricas basadas en encuestas ya que permiten ser exactas, permiten obtener una visión holística del sistema, y triangularlas con las basadas en sistemas.
2. Obtención rigurosa de umbrales. Un tema importante es el cálculo de los umbrales para las métricas, que puede realizarse de varias maneras estadísticas. Boucher y Badri (2018) investigan en este sentido la

utilización de curvas ROC (*Receiver Operating Characteristic*), el VARL (*Value of an Acceptable Risk Level*) y Alves rankings.

3. Procedimiento completo de preprocesamiento y validación de los datos recogidos. Trendowicz (2013) señala la necesidad de preprocesar los datos de la medición (lo que puede conllevar: formateo de datos, cambios en el formato de representación, concatenación o separación, etc.), integración de datos (combinación de datos, eliminación de redundancia, resolución de conflictos de datos), y limpieza de datos (incluyendo la gestión de los datos ausentes y de los *outliers*), y transformación de datos (normalización, discretización, conversión de unidades, agregación,) y la validación de los datos de medición (para asegurar su consistencia y corrección).
4. El programa de medición definido debe ser automatizado, aunque es preferible comenzar restringiendo a la utilización de herramientas específicas y sencillas antes que a herramientas complejas, ya que el personal debe inicialmente conocer bien el proceso de medición, sus objetivos y los pasos necesarios a seguir. A veces con el uso de herramientas complejas se pierde la metodología y se confunde a los usuarios. Estas herramientas pueden ser sencillas hojas de cálculo o bases de datos. Una vez que el proceso de medición esté establecido, se podrán utilizar otras herramientas más complejas.
5. Probar el programa de medición en un proyecto piloto antes de establecerlo en la organización. Aunque se trate de un programa de medición ya revisado, no es recomendable establecerlo en la organización para su uso generalizado hasta que no se haya probado en un proyecto piloto. El programa de medición debería ser verificado por el jefe del proyecto “piloto”, el grupo de analistas de medición y jefes de proyecto que cuenten con gran experiencia en otras áreas. La idea es utilizar el programa de medición en el proyecto, obtener los resultados y realizar los análisis para comprobar que el programa de medición es coherente, se ajusta al propósito, la información que se obtiene es útil, las herramientas de soporte están bien implementadas y son convenientes, etc. Si la prueba piloto resulta ser satisfactoria para el grupo de validación, se incluirá dentro del proceso de medición de la organización y se usará en los siguientes proyectos. Con los programas de medición enfocados en el proceso o en el producto, se hará también una prueba piloto haciendo un análisis previo al uso formal del programa de medición.
6. Supervisar a lo largo de todo el proceso de creación y funcionamiento del programa de medición, la calidad de los datos y de la información.
7. Evaluación continua de la robustez del programa de medición, para esto Staron y Meding (2016) proponen un método caracterizado por las siguientes cuestiones:
 - ¿Cómo se utilizan las métricas en la organización?
 - ¿Cómo resultan de útil las métricas para tomar decisiones?
 - ¿Cuál es la infraestructura de medición?
 - ¿Cómo de bien se aceptan las métricas en la organización? (madurez de medición)
 - ¿Cómo se recogen, analizan y visualizan las métricas?
 - ¿Cómo se identifican las buenas prácticas en organizaciones externas?
 - ¿Cómo se identifican las buenas prácticas de la investigación?

MÉTODO DE IMPLANTACIÓN DEL PROGRAMA DE MEDICIÓN

A continuación, se muestra una breve guía sobre cómo definir e implantar un programa de medición teniendo en cuenta los consejos y factores de éxito indicados anteriormente y lo expuesto en Díaz-Ley (2009) y Díaz-Ley et al. (2008).

Roles

Los diferentes roles que intervienen en la definición e implantación del programa de medición son los siguientes:

<https://dogramcode.com/programacion>

- **Grupo de definición:** está compuesto por uno o varios (dependiendo de la envergadura del programa de medición) analistas de medición y los promotores y algunos *stakeholders* del programa de medición. Para empresas pequeñas o medianas se recomienda que haya un solo analista de medición y una sola persona como interesado principal que debe conocer bien las necesidades del programa de medición, los proyectos que se llevan a cabo en la empresa, los procesos y productos. Sin embargo, puntualmente para ciertas consultas se pedirá ayuda otras personas relacionadas con el programa.
- **Grupo de revisores:** este grupo estará compuesto por el resto de *stakeholders* de los resultados del programa de medición y algunos desarrolladores que estarán involucrados en el proceso de medición. Los interesados (por ejemplo, los jefes de proyecto) revisarán si: el programa de medición cubre los objetivos requeridos; los análisis propuestos se pueden realizar tomando los datos de entrada especificados; los indicadores definidos cubren los objetivos del programa; los datos pueden ser recogidos de la manera propuesta, etc.
- **Grupo de aceptación:** en este grupo estarán todos o un grupo clave de *stakeholders* del programa de medición que finalmente lo aceptarán.

Metodología

Se pueden distinguir los siguientes pasos para llevar a cabo la definición e implantación del programa de medición:

- 1. Identificar los objetivos de mejora del proceso.** En caso de que la organización sea madura respecto a la medición del software, se podría desarrollar directamente un programa de medición partiendo de los objetivos de negocio.
- 2. Formalizar los objetivos de medición.** Se definen formalmente los objetivos de medición que dan soporte al objetivo de mejora del proceso o al objetivo de negocio. Además, se especifica el propósito, el enfoque, el punto de vista y el entorno donde se define este objetivo.
- 3. Definición de indicadores.** Puede ocurrir que la empresa tenga ya definido un programa de medición con un objetivo, contexto y propósito similar, por lo que se reutilizaría este modelo adaptándolo a las características específicas del proceso, producto o proyecto objetivo de medición. En caso contrario, se debe definir el indicador que da soporte al objetivo de medición, es decir, para cada objetivo de medición establecido, se detalla su correspondiente indicador completando la siguiente información: preguntas que se pretenden resolver, la representación gráfica del indicador, las entradas para generar el indicador, personas interesadas en la información resultante del indicador, personas responsables de generar, analizar e interpretar el indicador, información sobre cuándo se utilizarán los datos resultantes del indicador, la definición de los algoritmos necesarios para responder las preguntas y generar el indicador, las posibles salidas del indicador, posibles análisis e interpretaciones, etc.
- 4. Definición de indicadores derivados.** En el caso de que el indicador principal definido necesite otros indicadores más específicos para poder ser generado, se debe definir el objetivo de medición derivado y su correspondiente indicador. Es decir, se deben repetir las actividades dos y tres veces hasta que todos los objetivos de medición e indicadores estén definidos.
- 5. Definición de medidas y especificación de las necesidades para implementarlas.** Los indicadores definidos indican las medidas que se necesitan como entrada al indicador. En esta etapa se detalla la especificación de la medida a recolectar, es decir, qué datos se incluyen y cuáles se excluyen, cómo se obtendrán los datos y se analiza si es posible disponer de estos. En caso de que no sea posible obtenerlos, o bien se modifica el indicador que hace uso de estos datos y se analiza el impacto en el objetivo de medición final, o bien se busca la manera de poder obtenerlos. La segunda opción es la menos recomendable y la que hay que intentar evitar, ya que realizar cambios en el proceso o incluir herramientas para recolectar cierta

información puede interferir de manera substancial en el trabajo rutinario de los usuarios e, incluso, si estos cambios suponen una mejora, incrementa el riesgo de fracaso en la implantación del programa de medición.

6. **Integración del programa de medición.** En esta etapa se integra el programa de medición definido en el proceso de desarrollo. Por ejemplo, se pueden integrar en las plantillas de informes de seguimiento, los indicadores de monitorización definidos en el proyecto; en las plantillas de cierre de proyecto, los indicadores de calidad del desarrollo; en las actividades rutinarias de desarrollo, las actividades de recogida de datos, etc.
7. **Verificación del programa de medición.** El grupo de revisión verificará el programa de medición. Es conveniente que el analista de medición reúna al grupo para exponerle el programa de medición definido. Más tarde se deben dejar unos días para que lo revisen exhaustivamente y finalmente se realizará otra reunión para proponer los cambios necesarios. Tras estas reuniones el analista de medición actualizará el programa de medición.
8. **Automatización del programa de medición.** Tal vez sea necesario automatizar ciertas consultas para facilitar la captura de datos y reunir la información de distintas herramientas para poder realizar análisis globales. La automatización se puede realizar *ad-hoc* o adquiriendo una herramienta.
9. **Aceptación del programa de medición.** En esta etapa se prueba el programa de medición en un caso real, es decir, se utiliza el programa de medición en un proyecto piloto, o en un proceso o producto. Los resultados del programa de medición se presentan a los interesados (grupo de aceptación) para que evalúen si la información que se obtiene es coherente, útil, permite realizar los análisis requeridos etc. En caso afirmativo, el programa de medición se establece en la organización o unidad de negocio. Si el uso del programa de medición es puntual y útil solo para un proyecto o producto, esta etapa no es necesaria realizarla.

LECTURAS RECOMENDADAS

- Menzies, T., Williams, L. y Zimmermann, T. (eds.) (2016). *Perspectives on Data Science for Software Engineering*. Morgan Kaufmann, Amsterdam, Países Bajos.

En este libro se recopilan diversas propuestas sobre la recolección y el análisis de datos y la utilización de técnicas de la Ciencia de los Datos en la Ingeniería del Software.

- Staron, M. y Meding, W. (2018). *Software Development Measurement Programs. Development, Management and Evolution*. Cham, Suiza, Springer.

Estos conocidos expertos en el campo de la medición, abordan los diferentes aspectos, técnicos y organizativos, para poner en marcha un programa de medición de software.

Anexo A

MÉTODO PARA LA CREACIÓN Y VALIDACIÓN DE MÉTRICAS

La mayoría de las métricas propuestas han fracasado, debido generalmente a que carecían de atributos necesarios para ser válidas y útiles para el propósito para el que fueron creadas. La creación y validación de una métrica debe seguir un proceso metodológico que permita obtener medidas adecuadas.

Desde los años setenta han aparecido un gran número de métricas para capturar atributos del software de una forma cuantitativa. Sin embargo, muy pocas han sobrevivido con éxito la fase de definición y han resultado útiles. Esto se debe a múltiples problemas relativos a la validez teórica y empírica de muchas métricas, algunos los cuales se detallan a continuación (Briand *et al.*, 1999):

- Las métricas no se definen siempre en un contexto en el que sea explícito y esté bien definido su objetivo de interés. Por ejemplo: la reducción del esfuerzo de desarrollo o la reducción de los fallos presentes en los productos software.
- Incluso si el objetivo es explícito, las hipótesis experimentales, a menudo no se hacen explícitas, por ejemplo. ¿Qué se pretende deducir del análisis? ¿Es creíble el resultado?
- Las definiciones de las métricas no siempre tienen en cuenta el entorno o contexto en el que serán aplicadas, por ejemplo ¿Se puede utilizar una métrica definida para un entorno no orientado a objetos en un contexto orientado a objetos?
- No siempre es posible realizar una validación teórica adecuada de la métrica porque el atributo que queremos medir no siempre está bien definido, por ejemplo, la noción de complejidad.
- Un gran número de métricas nunca se han validado empíricamente, por ejemplo, ¿qué métrica de tamaño predice mejor el esfuerzo en un entorno de desarrollo?

Por ello en el grupo Alarcos hemos ido definiendo y refinando un método para la definición de métricas válidas (Calero *et al.*, 2001; Serrano *et al.*, 2002; Reynoso *et al.*, 2010) que se ha utilizado en varias tesis doctorales y proyectos de investigación; así como en la creación de AQCLab, el primer laboratorio acreditado por ENAC para la evaluación de la calidad del software².

A.1 MÉTODO DE TRABAJO

La definición de las métricas debe basarse en objetivos claros y siguiendo las necesidades de la organización. Teniendo en cuenta todas las consideraciones expuestas, en la Figura A.1 se muestra el proceso de definición de métricas, de manera que se puedan conseguir métricas válidas y útiles para productos software. En esta figura se puede observar que el método consta de diversas fases que van desde la identificación de los objetivos y las hipótesis de trabajo hasta la aplicación y posterior retirada de una métrica; las flechas continuas representan el flujo de las métricas y las discontinuas representan el flujo de información a lo largo de todo el proceso. Este proceso consta de cinco etapas principales:

- **Identificación:** Se definen los objetivos que se persiguen a la hora de crear la métrica y se plantean las hipótesis de cómo se llevará a cabo la medición. Sobre los elementos de esta etapa (objetivos e hipótesis) se basarán todas las etapas siguientes. Como resultado de esta etapa se generan los requisitos que debe cumplir la métrica.
- **Creación:** Se realiza la definición de la métrica y su validación teórica y empírica. Esta etapa es una de las más importantes y larga pues como abarca un proceso iterativo del que debe salir una métrica válida tanto formal como empíricamente.
- **Aceptación:** Una vez obtenida una métrica válida, suele ser necesario pasar por una etapa de aceptación de la métrica en la que se harán pruebas en entornos reales, de manera que podamos comprobar si la métrica cumple los objetivos deseados dentro del campo de aplicación real.

- **Aplicación:** Una vez que tengamos una métrica aceptada, la utilizaremos dentro del campo de la aplicación para la que fue diseñada.
- **Acreditación:** Es la última etapa del proceso, que discurre en paralelo con la fase de aplicación y tiene como objetivo el mantenimiento de la métrica, de manera que se pueda adaptarla al entorno cambiante de aplicación. Como consecuencia de esta etapa, puede que una métrica sea retirada, porque ya no sea útil en el entorno en el que se aplica o que se reutilizada para iniciar el proceso de nuevo.

A continuación, se detallan en la Figura A.1 las etapas que componen el método propuesto.

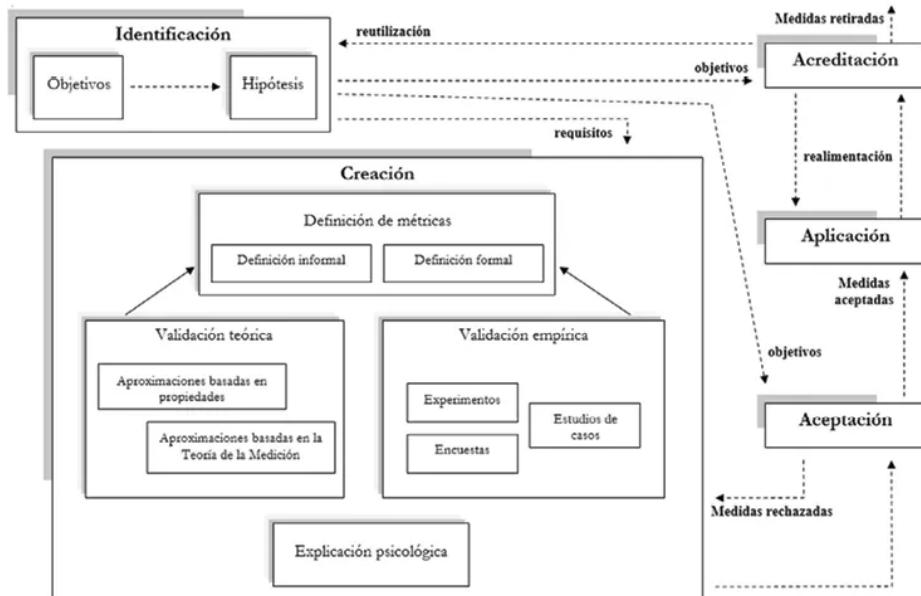


Figura A.1. Método de investigación para la definición de métricas

A.2 IDENTIFICACIÓN

En esta etapa se pretende identificar los objetivos de la métrica y las hipótesis en las que nos basamos para crearla. Los objetivos indican lo que se pretende conseguir con la utilización del proceso de métrica y representan la razón por la que se llevará el proceso de creación de la métrica (el “porqué”). Las hipótesis son la forma en la que se pretende llevar a cabo la medición (el “cómo”), identificando la información que se debe manejar para conseguir alcanzar los objetivos deseados. Este proceso suele estar basado en la experiencia y el conocimiento de los expertos y puede utilizar mecanismos basados en GQM.

Más concretamente, se llevan a cabo las siguientes actividades:

- Seleccionar la entidad de estudio, es decir, el producto, proceso, proyecto o recurso que se pretende caracterizar midiendo sus atributos.
- Determinar el foco de la calidad, es decir, los atributos en los cuales centraremos las métricas, por ejemplo, mantenibilidad, portabilidad, etc.
- Determinar los objetivos GQM a nivel conceptual; Analizar el “objeto del estudio” con el “propósito” respecto al “foco de calidad” desde el “punto de vista”.
- Determinar las propiedades estructurales a estudiar, por ejemplo, acoplamiento, cohesión, tamaño, etc.
- Identificar las abstracciones para medir las propiedades estructurales, por ejemplo, si se trata del acoplamiento identificar los diferentes tipos de conexiones que constituyen acoplamiento, el lugar del impacto del acoplamiento, su granularidad, etc.
- Refinar los objetivos en preguntas a nivel operacional.

Como resultado de esta fase se deben obtener los requisitos que debe cumplir la métrica, que serán utilizados en la etapa de creación. Además, como se observa en la Figura A.1, los objetivos serán utilizados en las etapas de aceptación, aplicación y acreditación.

A.3 CREACIÓN

El proceso de creación es aquel en el que a partir de los requisitos obtenidos en la etapa de identificación se creará una métrica válida, lista para ser aplicada en entornos reales. Como se puede observar en la Figura A.1, el proceso de creación de las métricas es evolutivo e iterativo y se subdivide en varias etapas intermedias. Como resultado de la retroalimentación, las métricas deben ser redefinidas de acuerdo a las validaciones, teóricas o empíricas, fallidas.

Al final de la etapa de creación, las métricas se considerarán válidas y aquellas que no sean válidas, serán descartadas.

A.3.1 Definición

Es el primer paso de esta fase que debe realizarse considerando las características del producto que vamos a medir y la experiencia de los profesionales. Es recomendable conseguir esta definición de una forma metodológica, aplicando, por ejemplo, la aproximación GQM en el que se puede definir el propósito de la métrica que se define, el punto de vista del usuario y el contexto de uso. También puede ser aconsejable utilizar algún metamodelo para definir las métricas.

En la definición se deben considerar objetivos claros, es decir, realizar una definición de la métrica orientada al objetivo para evitar obtener una métrica que no cumple con el objetivo deseado.

Por otro lado, aunque en una primera etapa definamos la métrica en lenguaje natural, es deseable que la definición de las métricas se realice de manera formal para evitar ambigüedades.

A.3.2 Validación teórica

El objetivo principal de la validación teórica es comprobar si la idea intuitiva acerca del atributo que está siendo medido se refleja en la métrica. Esto se hace analizando los requisitos que deben ser satisfechos cuando estamos midiendo. Además, la validación teórica proporciona información relacionada con las operaciones matemáticas y estadísticas que pueden ser realizadas con la métrica, lo cual es esencial cuando tengamos que trabajar con ella.

Lamentablemente no existe un estándar para la validación formal a través del cual obtener la información matemática de las métricas definidas, sin embargo, hay dos tendencias principales en la validación: los marcos basados en aproximaciones axiomáticas (que definen formalmente propiedades deseables de las métricas para un atributo software concreto) (Briand *et al.*, 1996b; Weyuker, 1988) y los que se basan en la teoría de la métrica (Whitmire, 1997; Zuse, 1998; Poels y Dedene, 2000) cuyo objetivo es obtener la escala matemática a la que pertenece una métrica, y por tanto sus transformaciones admisibles, estadísticos y test aplicables y especifican un marco general en el que las métricas deben ser definidas.

A.3.3 Validación empírica

El objetivo de esta etapa es probar la utilidad práctica de las métricas propuestas utilizando encuestas, experimentos y estudios de casos.

A.3.4 Explicación psicológica

Idealmente deberíamos ser capaces de explicar la influencia de los valores de las métricas desde un punto de vista psicológico. Por ejemplo, utilizando el concepto de complejidad cognitiva, entendida como el esfuerzo mental de una persona que está tratando con un artefacto software. Esto lleva consigo dos ventajas principales: es útil para definir la razón de cada definición de métrica, ya que muchas métricas se relacionan con limitaciones cognitivas

(Klemola y Rilling, 2003) y además nos proporciona un conocimiento muy importante a la hora de explicar los hallazgos experimentales.

A.4 ACEPTACIÓN

Suele ser necesaria la existencia de una fase de pruebas en la que se realice una experimentación sistemática con usuarios reales para verificar si cumplen los objetivos buscados dentro de un entorno de trabajo real (esta etapa se diferencia de los estudios de casos en que en éstos últimos no se suele trabajar en el entorno final de aplicación). En definitiva, intenta encontrar si las métricas “válidas” que se consiguieron al final de la fase de creación son aceptables en entornos de aplicación reales, teniendo en cuenta los objetivos obtenidos en la etapa de identificación.

Esta etapa debe ser realizada con proyectos no críticos y con riesgos controlados. Idealmente debería usarse en proyectos piloto de manera que el fracaso de aceptación de la métrica no suponga un fracaso en un proyecto importante.

Si se demuestra que la métrica sigue cumpliendo los objetivos, estaremos en disposición de pasar a la etapa de aplicación, si no es así, deberemos volver a la etapa de creación.

A.5 APPLICACIÓN

En esta etapa utilizaremos la métrica ya aceptada en el entorno real. Esta fase discurrirá en paralelo con la fase de acreditación.

A.6 ACREDITACIÓN

Esta última fase del proceso es una etapa dinámica que persigue el aseguramiento de la métrica y la mejora continua de la misma, en función de cómo evoluciona el entorno de aplicación, de manera que podamos seguir cumpliendo los objetivos que se perseguían al principio del método.

En ocasiones el entorno puede variar tanto (por ejemplo, pasar de un entorno estructurado a uno orientado a objetos) que la métrica no sea aplicable, en este caso, la métrica debería ser descartada y el conocimiento adquirido durante su tiempo de vida debería realimentarse a la etapa de identificación de manera que podamos crear una métrica adecuada para el nuevo entorno cumpliendo los objetivos perseguidos. Además, al utilizar la experiencia de la utilización de la métrica descartada, tendremos más probabilidades de formular hipótesis correctas en la etapa de identificación.

A.7 LECTURAS RECOMENDADAS

- *Métodos de Investigación en Ingeniería del Software. 2^a ed. Genero, M., Cruz-Lemus, J.A., Díaz, O. Y Piattini, M., 2020, Madrid, Ra-Ma*

Este libro ofrece una visión general de la utilización de las encuestas, experimentos, estudios de caso e investigación en acción en la Ingeniería del Software, que permiten validar métricas de manera empírica.

² <http://www.aqclab.es>

ACRÓNIMOS

- **AC.** Afferent Couplings
- **ACM.** Association for Computing Machinery
- **ADM.** Architecture Driven Modernization

- **AENOR.** Asociación Española de Normalización y Certificación
- **BPR.** Business Process Reengineering
- **CBO.** Coupling Between Object classes
- **CEM.** Centro Español de Metrología
- **CHP.** Coste por Hora-Programador
- **CISQ.** Consortium for IT Software Quality
- **CLOC.** Commented Lines of Code
- **CMM.** Capability Maturity Model
- **CMMI.** Capability Madurity Model Integration
- **CMMI-DEV.** CMMI for Development
- **CMMI-PPL.** CMMI for People Management
- **CMMI-SPM.** CMMI for Supplier Management
- **CMMI-SVC.** CMMI for Services
- **COCOMO.** Constructive Cost Model
- **COSMOS.** Cost Management with Metrics of Specification
- **COSMIC.** Common Software Metrics Internaciona Consortium
- **COTS.** Commercial Off-The-Shelf
- **CPI.** Cost Performance Index
- **CSCW.** Computer-Supported Cooperative Work,
- **CTP.** Coste total actual del proyecto
- **DAC.** Data Abstraction Coupling
- **DEVOPS .** Development and Operations
- **DIT.** Depth of Inheritance Tree
- **DVCS.** Distributed Version Control Systems
- **ECC.** External Class Complexity
- **ECS.** External Class Size
- **ENAC.** Entidad Nacional de Acreditación
- **ESEIW.** Empirical Software Engineering International Week
- **ESEM.** Empirical Software Engineering and Measurement
- **FCE.** Factores Críticos de Éxito
- **FISMA FSM.** Finnish Software Measurement Association
- **FSM.** Functional Size Measurement Method
- **GDSM.** Goal-Driven Software Measurement
- **GQ(I)M.** Goal Question Indicator Metric
- **GQM.** Goal Question Metric
- **GQM/MEDEA.** GQM - Metric Definition Approach
- **GQM+S.** GQM+Strategies

- **HPD.** Horas-programador diarias
- **HPT.** Horas programador totales
- **IEC.** International Electrotechnical Commission
- **IEEE.** Institute of Electrical and Electronics Engineers
- **IFPUG.** International Function Point Users Group
- **ISESE.** International Symposium on Experimental Software Engineering
- **ISO.** International Organisation for Standardization
- **KPI.** Key Performance Indicator
- **LCOM.** Lack of Cohesion in Methods
- **LOC.** Lines of Code
- **MERMAID.** Metrication and Resource Modelling Aid
- **METKIT.** Metrics Educational Toolkit
- **MOF.** Meta-Object Facility
- **MOOSE.** Metrics for Object-Oriented Software Engineering
- **MPC.** Message Passing Coupling
- **MPM.** Managing Performance and Measurement
- **MSS.** Measurement Support System
- **MTTD.** Mean Time To Defect
- **NASA.** National Aeronautics and Space Administration
- **NCLOC.** Non Comment Lines of Code
- **NOC.** Number of Children
- **NOM.** Number of Methods
- **OMG.** Object Management Group
- **OO.** Orientación a Objetos
- **OTSO.** Off-The-Shelf Option
- **PDCA.** Plan, Do, Check, Act
- **PMBOK.** Project Management Body of Knowledge
- **PMI.** Project Management Institute
- **PSM.** Practical Software Measurement
- **PSP.** Personal Software Process
- **QM-RM.** Quality Measurement Reference Model
- **QM.** Quality Measures
- **QME.** Quality Measure Element
- **RFC.** Response for a Class
- **ROC.** Receiver Operating Characteristic
- **SDI.** System Design Stability
- **SEI.** Software Engineering Institute

- **SLA.** Service Level Agreement
- **SLIM.** Software Lifecycle Management
- **SMM.** Structured Metrics Meta-Model
- **SPC.** Statistical Process Control
- **SPI.** Schedule Performance Index
- **TAME.** Tailoring a Measurement Environment
- **TCC.** Tight Class Cohesion
- **TI.** Tecnologías de la Información
- **TSP.** Team Software Process
- **VARL.** Value of an Acceptable Risk Level
- **V-GQM.** Validating Goal Question Metric
- **VIM.** International Vocabulary of Metrology
- **WMC.** Weighted Methods per Class

REFERENCIAS

- Al-Nanih, R., Al-Nuaim, H. y Ormandjieva, O. (2009). New Health Information Systems (HIS) quality-in-use model based on the GQM approach and HCI principles. 13th International Conference on Human-Computer Interaction, HCI International 2009, San Diego, CA; EE.UU.
- Albrecht, A. (1979). Measuring Application Development Productivity. Paper presented at the Proceedings of the IBM Application Development Symposium, Monterey.
- Arvanitou, E.M., Ampatzoglou, A., Chatzigeorgiou, A., Galster, M. y Avgeriou, P. (2017). A mapping study on design-time quality attributes and metrics. *The Journal of Systems and Software* 127, 52-77.
- Aslan, D., Tarhan, A. y Demirörs, V.O. (2014). How process enactment data affects product defectiveness prediction-a case study (Conference Paper) *Studies in Computational Intelligence*, 496, 151-166.
- Assila, A., De Oliveira, K.M. y Ezzedine, H. (2014). Towards quality indicators for supporting the evaluation of interactive systems HCI. 14th Conference on Ergonomics and Advanced Informatics - Design, Ergonomics and Human-Computer Interaction: What Integration for Interaction Co-design, ErgoIA 2014.
- Assila, A., Marçal de Oliveira, K. y Ezzedine, H. (2016). Integration of Subjective and Objective Usability Evaluation Based on ISO/IEC 15939: A Case Study for Traffic Supervision Systems. *International Journal of Human-Computer Interaction*, 32 (12), 931-955
- Assila, A., Plouzeau, J., Merienne, F., Erfanian, A. y Hu, Y. (2017). Defining an indicator for navigation performance measurement in VE based on ISO/IEC15939. 4th International Conference on Augmented Reality, Virtual Reality and Computer Graphics, SALENTO AVR 2017; Ugento; Italy; 12 June 2017 through 15 June 2017, Volume 10324 LNCS 10324,17-34.
- Aversano, L., Bodhuin, T., Canfora, G. y Tortorella, M. (2004). A Framework for Measuring Business Processes Based on GQM. Paper presented at the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 1.
- Baker, A., Bieman, D., Gustafson, D. y Melton, C. (1987). Modeling and measuring the software development process. Paper presented at the Proceedings 12th Hawai International Conference on System Sciences.

- Baldassarre, M.T., Boffoli, N., Caivano, D. y Visaggio, G. (2004). Managing Software Process Improvement (SPI) through Statistical Process Control (SPC). Paper presented at the PROFES 2004.
- Baldassarre, M.T., Caivano, D. y Visaggio, G. (2003). Comprehensibility and Efficiency of Multiview Framework for Measurement Plan Design. Proceedings of the 2003 International Symposium on Empirical Software Engineering.
- Barcellos, M.P., de Almeida Falbo, R., y Rocha, A.R. (2013). A strategy for preparing software organizations for statistical process control. *Journal of the Brazilian Computer Society*, 19(4), 445-473.
- Basili, V., Briand, L., Condon, S., Kim, Y., Melo, W. y Valett, J. (1996). Understanding and predicting the process of software maintenance release. Paper presented at the Proceedings of the 18th International Conference on Software engineering ICSE '96.
- Basili, V., Lampasona, C. y Ocampo, A.E. (2013). Aligning Corporate and IT Goals and Strategies in the Oil and Gas Industry. Paper presented at the 14th International Conference, PROFES 2013, Paphos, Cyprus.
- Basili, V., McGarry, F., Pajerski, R. y Zelkowitz, M.V. (2002). Lessons learned from 25 years of process improvement: the rise and fall of the NASA software engineering laboratory. Paper presented at the Proceedings of the 24th International Conference on Software Engineering ICSE '02.
- Basili, V. y Rombach, D. (1987). TAME: Integrating Measurement into Software Environments, Department of Computer Science, University of Mariland, 31.
- Basili, V. y Rombach, D. (1988). The Tame Project: Towards Improvement-Oriented Software Environments. *IEEE Transactions of Software Engineering* 14, 6: 758-773.
- Basili, V., Selby, W. y Hutchens, D. (1986). Experimentation in software engineering. *IEEE Transactions on Software Engineering*, 12(7), 733-743.
- Basili, V., Shull, F. y Lanubile, F. (1999). Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, 25(4), 435-437.
- Basili, V., Trendowicz, A., Kowalczyk, M., Heidrich, J., Seaman, C., Münch, J. y Rombach, D. (2014). Aligning Organizations Through Measurement. The GQM+ Strategies approach: Springer.
- Basili, V. y Weiss, D. (1984). A Methodology for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering*, 10, 11, 758-773.
- Bassman, M.J., McGarry, F. y Pajerski, R. (1995). Software measurement guide-book: NASA Goddard Space Flight Center, Software Engineering Laboratory.
- Belady, L. (1981). Complexity of large systems. In A. Perlis, F. Sayward & M. Shaw (Eds.), *Software Metrics* (225-233). Cambridge: MIT Press.
- Benedicenti, L., Succi, G., Valerio, A. y Vernazza, T. (1996). Monitoring the efficiency of a reuse program. *ACM SIGAPP Applied Computing Review*, ACM Press, 4(2), 8-14.
- Berander, P. y Jönsson, P. (2006). Metrics and measurement: A goal question metric based approach for efficient measurement framework definition. Paper presented at the Proceedings of the 2006 ACM/IEEE international symposium on International symposium on empirical software engineering ISESE '06.
- Berry, M., Jeffery, R. y Aurum, A. (2004). Assessment of Software Measurement: an Information Quality Study. Paper presented at the Proceedings of the 10th International Symposium on Software Metrics (METRICS'04).
- Bhandari, I.S., Mendonca, M.G. y Dawson, J. (1995). On the use of machine-assisted knowledge discovery to analyze and reengineer measurement frameworks. Paper presented at the Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research CASCON '95.
- Bieman, J.M. (1991). Deriving Measures of Software Reuse in Object Oriented Systems.

- Birk, A., Solingen, R.V. y Järvinen, J. (1998). Business Impact, Benefit, and Cost of Applying GQM in Industry: An Indepth, Long-term Investigation at Schlumberger RPS. Paper presented at the Proceedings of the 5th International Symposium on Software Metrics (METRICS'98), Bethesda Maryland.
- Boehm, B. (1981). Software Engineering Economics. Englewood Cliffs, Nueva Jersey: Prentice Hall.
- Bonifati, A., Cattaneo, F., Ceri, S., Fuggetta, A. y Paraboschi, S. (2001). Designing data marts for data warehouses. ACM Transactions on Software Engineering and Methodology (TOSEM), 10(4), 452-483.
- Boucher, A. y Badri, M. (2018). Software metrics thresholds calculation techniques to predict fault-proneness: An empirical comparison. Information and Software Technology 96, 38–67.
- Briand, L., Differding, C.M. y Rombach, H.D. (1996a). Practical Guidelines for Measurement-Based Process Improvement. Software Process - Improvement and Practice, 2(4), 253-280.
- Briand, L., Morasca, S. y Basili, V. (1996b). Property-Based Software Engineering Measurement. IEEE Transactions on Software Engineering, 22(1), 68-86.
- Briand, L., Arisholm, S., Counsell, F., Houdek, F. y Thévenod-Fosse, P. (1999). Empirical Studies of Object-Oriented Artifacts, Methods, and Processes: State of the Art and Future Directions. Empirical Software Engineering, 4(4), pp. 387-404.
- Briand, L., Morasca, S. y Basili, V. (2002). An Operational Process for Goal-Driven Definition of Measures. IEEE Transactions on Software Engineering, 28, 1106-1125.
- Brito, D., Barcellos, M. y Santos, G. (2018). Investigating measures for applying statistical process control in software organizations. Journal of Software Engineering Research and Development, pp. 6-10.
- Brito e Abreu, F. y Carapuça, R. (1994). Object-Oriented Software Engineering: Measuring and controlling the development process. Paper presented at the Proceedings of the 4th International Conference on Software Quality, McLean (USA).
- Caballero, I., Pérez, R., Calero, C. y Piattini, M. (2009). MEPLAMECAL: A methodology based on ISO/IEC 15939 to elaborate data quality measurement plans 13th Conference on Software Engineering and Databases, JISBD'08. IEEE Latin America Transactions 7 (3), 361-368.
- Caivano, D. (2005). Continuous software process improvement through statistical process control. Paper presented at the In Proceedings of the Ninth European Conference on Software Maintenance and Reengineering (CSMR'05), Manchester, UK.
- Calero, C., Piattini, M. y Genero, M. (2001). Empirical Validation of Referential Integrity. Information and Software Technology, 43, 949-957.
- Canedo, E., Valenca, K. y Santos, G. (2019). An Analysis of Measurement and Metrics Tools: A Systematic Literature ReviewProceedings of the 52nd Hawaii International Conference on System Sciences, 6970-6980.
- Carbone, M., y Santucci, G. (2002). Fast && Serious: a UML-Based Metric for Effort Estimation. Paper presented at the Proceedings of the 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2002), Malaga, Spain.
- Card, D. (2003). Integrating Practical Software Measurement and the Balanced Scorecard. Paper presented at the Proc. of the 27th Annual International Computer Software and Applications Conference (COMPSAC'03).
- Card, D. y Agresti, W. (1988). Measuring Software Design Complexity. Journal Sstems and Software, 8, 185-197.
- Card, D. y Berg, R.A. (1989). An Industrial Engineering Approach to Software Development. J. Systems and Software, 10, 159-168.
- Carvalho, R. M., Andrade, R. M. C. y Oliveira, K. M. (2015). Using the GQM Method to Evaluate Calmness in Ubiquitous Applications. In N. Streitz & P. Markopoulos (eds.), Third International Conference Distributed, Ambient, and Pervasive Interactions:, DAPI 2015. Springer International Publishing.

- CMMI (2018). CMMI V2.0 Development Model. CMMI Institute.
- Cohn, M. (2005). Agile Estimation and Planning. Pearson Education
- Coulter, N. (1983). Software Science and Cognitive Psychology. IEEE Transactions on Software Engineering, 9(2), 166-171.
- Curtis, B. (1979). In Search of Software Complexity. Paper presented at the En Workshop on Quantitative Software Models for Reliability.
- Curtis, B. (1981). Experimental Evaluation of Software Characteristics. In A. Perlis, F. Sayward & M. Shaw (Eds.), Software Metrics (pp. 62-75). Cambridge: MIT Press.
- Chang, C.-W. y Tong, L-I. (2013). Monitoring the software development process using a short-run control chart. Software Quality Journal, 21(3), 479-499.
- Chapin, N. (1979). A Measure of Software Complexity. Paper presented at the Proceedings of the AFIPS National Computer Conference.
- Chidamber, S. y Kemerer, C. (1994). A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering, 20(6), 476-493.
- Daskalantonakis, M.K. (1992). A Practical View of Software Measurement and Im-plementation Experiences Within Motorola. IEEE Transactions on Software Engineering, 18(11), 998-1010.
- Daskalantonakis, M.K., Yacobellis, R.H. y Basili, V. (1990). A Method for Assessing Software Measurement Technology. Quality Engineering, 27-40.
- Davis, C.W.H. (2015). Agile Metrics in Action. How to measure and improve team performance. Shelter Island, NY: Manning Publications, Co.
- DeMillo, R. y Lipton, R. (1981). Software Project Forecasting. In A. Perlis, F. Sayward & M. Shaw (Eds.), Software Metrics, Cambridge, MIT Press, 77-89.
- De Oliveira, K.B. y Belchior, A.D. (2006). Institutionalization of an organizational measurement process. International Conference on Systems, Computing Sciences and Software Engineering, SCSS 2006, 253-258.
- Desouza, K.C. (2015). Creating a Balanced Portfolio of Information Technology Metrics. IBM Center for the Business of Government.
- Díaz-Ley, M. (2009). Measurement Framework for the Definition of Software Measurement Programs in SMEs: MIS-PyME. Tesis Doctoral.: Dpto. Tecnologías y Sistemas de Información. Universidad de Castilla-La Mancha.
- Díaz-Ley, M., García, F. y Piattini, M. (2008). Implementing a software measurement program in small and medium enterprises: a suitable framework. IET Software, 2(5), 417-436.
- Díaz-Ley, M., García, F. y Piattini, M. (2010). MIS-PyME software measurement capability maturity model - Supporting the definition of software measurement programs and capability determination. Advances in Engineering Software, 41(10-11), 1223-1237.
- DoD. (2000). PSM: Practical Software and Systems Measurement - A Foundation for Ob-jective Project Management (Vol. Version 4.0c): Department of Defense and US Army.
- DoD. (2003). Practical Software and Systems Measurement. A Foundation for Objective Project Management. Version 4.0c. .
- Ebert, C., Dumke, R., Bundschuh, M. y Schmietendorf, A. (2004). Best Practices in Software Measurement: SpringerVerlag.
- Evangelist, M. (1984). An analysis of control flow complexity. Paper presented at the Proceedings of Annual International Computer Software and Applications Conference (COMPSAC).
- Fenton, N. (1991). Software Metrics: A Rigorous Approach: Chapman & Hall.

- Fenton, N. (2001). Metrics for Software Process Improvement. In H. M., O. E. W. & B. L. (Eds.), *Software Process Improvement: Metrics, Measurement and Process Modelling* (pp. 34-55): Springer.
- Fenton, N. y Bieman, J. (2014). *Software Metrics: A Rigorous and Practical Approach*, Third Edition: Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series.
- Fenton, N. y Melton, A. (1990). Deriving structurally based software measures. *Journal of Systems and Software*, 12, 177-187.
- Fenton, N. y Pfleeger, S.L. (1997). *Software Metrics: A Rigorous & Practical Approach* (2^a ed.), PWS Publishing Company.
- Fernandez-Corrales, C., Jenkins, M. y Villegas, J. (2013). Application of statistical process control to software defect metrics: An industry experience report. Paper presented at the International Symposium on Empirical Software Engineering and Measurement.
- Feyh, M. y Petersen, K. (2013). Lean software development measures and indicators - A systematic mapping study. 4th International Conference on Lean Enterprise Software and Systems, LESS 2013. Lecture Notes in Business Information Processing, 167, 32-47.
- Florac, W. y Carleton, A.D. (1999). *Measuring the Software Process. Statistical Process Control for Software Process Improvement*: Addison Wesley.
- Florac, W.A. (1992). *Software Quality Measurement: A Framework for Counting Problems and Defects*. Carnegie Mellon University, Pittsburgh, Pa: Software Engineering Institute.
- Florence, A. (2001). CMM Level 4 Quantitative Analysis and Defect Prevention. *CrossTalk*.
- Forsgren, N. y Kersten, M. (2018). DevOps metrics. *Commun. ACM* 61(4) 44-48.
- Fuggetta, A., Lavazza, L., Morasca, S., Cinti, S., Oldano, G. y Orazi, E. (1998). Applying GQM in an industrial software factory. *ACM Trans. Softw. Eng. Methodol*, 7(4), 411-448.
- García, F., Bertoá, M., Calero, C., Vallecillo, A., Ruiz, F., Piattini, M. y Genero, M. (2005). Towards a Consistent Terminology for Software Measurement. *Information and Software Technology*, 48(8), 631-644.
- Gartner. (2015). Market Trends: DevOps — Not a Market, but a Tool-Centric Philosophy That Supports a Continuous Delivery Value Chain. Disponible en: <http://www.gartner.com/document/2987231>.
- Gencel, C., Petersen, K., Mughal, A.A. y Iqbalb, M.I. (2013). Decision support framework for metrics selection in goal-based measurement programs: GQM-DSFMS. *The Journal of Systems and Software*, 86, 3091-3108.
- Geppert, B. y Weiss, D.M. (2003). Goal-Oriented Assessment of Product-Line Domains. Paper presented at the 9th International Software Metrics Symposium (METRICS'03).
- Gilb, T. (1977). *Software Metrics*. Cambridge, Massachusetts: Winthrop Publishers.
- Goethert, W. (1992). *Software Effort Measurement: A Framework for Counting Staff- Hours*. Pittsburgh, EE.UU. Software Engineering Institute, Carnegie Mellon University.
- Goethert, W. y Siviy, J. (2004). Applications of the Indicator Template for Measurement and Analysis: Software Engineering Institute.
- Gresse, C., Punter, T. y Anacleto, A. (2003). Software measurement for small and medium enterprises. Paper presented at the 7th International Conference on Empirical Assessment in Software Engineering (EASE), Keele, UK.
- Halstead, M. (1977). *Elements of Software Science*. New York: Elsevier North Holland.
- Hall, T. y Fenton, N. (1997). Implementing Effective Software Metrics Programs. *IEEE software*, 14(2), 55-65.
- Heidrich, J. y Trendowicz, A. (2011). Aligning Software Projects with Business Objectives. Nara, Japón.
- Helmer-Heidelberg, O. (1966). *Social Technology*. New York: Basic Books.

- Henry, S. y Kafura, S. (1981). Software Structure Metrics Based on Information Flow. IEEE Transactions on Software Engineering, 7(5), 510-518.
- Houdek, F. y Kempter, H. (1997). Quality patterns—an approach to packaging software engineering experience. ACM SIGSOFT Software Engineering Notes. Paper presented at the Proceedings of the 1997 symposium on Software reusability SSR '97.
- Hussain, A. y Ferneley, E. (2008). Usability Metric for mobile application: A goal question metric (GQM) approach (Conference Paper) Paper presented at the Proceedings of the 10th International Conference on Information Integration and Web-Based Applications and Services, IIWAS 2008, Linz; Austria.
- IEEE (1989). Guide for the Use of Standard Dictionary of Measures to Produce Reliable Software (Corrected Edition ed.): IEEE Standard Board.
- Ishigaki, D. y Jones, C. (2003). Practical Measurement in the Rational Unified Process. The Rational Edge. from <http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jano3/PracticalMeasurementInRUP>
- ISO (2017a). ISO/IEC 12207 Systems and software engineering - Software life cycle processes. Ginebra: International Organization for Standardization.
- ISO (2017b). ISO/IEC 15939. Software Engineering - Software Measurement Process. Ginebra, Suiza: International Organization for Standardization.
- ISO (2019). ISO/IEC 25021. Systems and Software Engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality Measures Element. Ginebra: International Organization for Standardization.
- Jalote, P. (1999). CMM in Practice: Processes for Executing Software Projects at Infosys: Addison-Wesley.
- Juristo, N. y Moreno, A. (2001). Basics of Software Engineering Experimentation: Kluwer Academic Publishers.
- Khomh, F., Vaucher, S., Guéhéneuc, Y.-G. y Sahraoui, H. (2011). BDTEX: A GQM-based Bayesian approach for the detection of antipatterns. Journal of Systems and Software, 84(4), 559-572.
- Klemola, T. y Rilling, J. (2003). A Cognitive Complexity Metric Based on Category Learning. Proceedings of the 2nd IEEE International Conference on Cognitive Informatics (ICCI '03). IEEE Computer Society, Washington, DC, EE.UU., 106.
- Komi-Sirviö, S., Parviainen, P. y Ronkainen, J. (2001). Measurement Automation: Methodological Background and Practical Solutions- A Multiple Case Study. Paper presented at the Proceedings of the Seventh International Software Metrics Symposium(Metrics '01).
- Komuro, M. (2006). Experiences of applying SPC techniques to software development processes. Proceedings of the 28th International Conference on Software Engineering, Shanghai, China.
- Kontio, J., Caldiera, G. y Basili, V. (1996). Defining factors, goals and criteria for reusable component evaluation. Paper presented at the Proceedings of the 1996 conference of the Centre for Advanced Studies on Collaborative research CASCON '96.
- Kupiainen, E., Mäntylä, M. y Itkonen, J. (2015). Using metrics in Agile and Lean Software Development - A systematic literature review of industrial studies. Information & Software Technology, 62, 143-163.
- Kurtel, K. y Ozemre, M. (2013). Cohesive software measurement planning framework using ISO standards: A case study from logistics service sector. Journal of software: Evolution and Process, 25 (7), 663-679.
- Kurtel, K. (2013). Measuring and monitoring software maintenance services: An industrial experience. Vianden, M., Licher, H. y Steffens, A. (2013). Towards a maintainable federalist enterprise measurement infrastructure. Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement, IWSM-MENSURA 2013, 247-252.
- Lantzy, M.A. (1992). Application of Statistical Process Control to Software Processes. Paper presented at the WADAS '92. Proceedings of the Ninth Washington Ada Symposium on Empowering Software Users and <https://dogramcode.com/programacion>

Developers.

- Lassez, J.L., van der Knijff, D., Sheperd, J. y Lassez, C. (1981). A critical examination of software science. *Journal of Systems and Software*, 2, 105-112.
- Lavazza, L. (2000). Providing Automated Support for the GQM Measurement Process. *IEEE software*, 17(3), 56-62.
- Lavazza, L. y Mauri, M. (2006). Software Process Measurement in Real World: Dealing with Operating Constraints. Paper presented at the Workshop on Software Process Simulation and Modeling.
- Lavazza, L., Frumento, E. y Mazza, R. (2015). Defining and evaluating software project success indicators: A GQM-based case study. 10th International Joint Conference on Software Technologies (ICSOFT), Colmar, 2015, pp. 1-12.
- Lee, D.H., y Park, J.J. (2012). Big data analysis of software performance trend using SPC with flexible moving window and fuzzy theory. *Journal of Institute of Control, Robotics and Systems*, 18(11), 997-1004.
- Li, W. y Henry, S. (1993). Object-Oriented metrics that predict maintainability. *Journal of Systems and Software*, 23(2), 111-122.
- Lindvall, M., y Rus, I. (2000). Process diversity in software development. *Software*, IEEE, 17(4), 14-18.
- Lindvall, M., Tesoriero, R., y Costa, P. (2002). Avoiding Architectural Degeneration: An Evaluation Process for Software Architecture. Paper presented at the 8th IEEE Symposium on Software Metrics (METRICS'02).
- López, G., Aymerich, B., Garbanzo, D. y Pacheco, A. (2016). Application of GQM+Strategies in a Multi-industry State-Owned Company. 17th International Conference Product-Focused Software Process Improvement:, PROFES 2016, Trondheim, Norway, pp. 198-214. Springer International Publishing.
- Lorenz, M. y Kidd, J. (1994). Object-Oriented Software Metrics: A Practical Guide. Englewood Cliffs (Nueva Jersey): Prentice Hall.
- Luo, Y. y van den Brand, M. (2016). Metrics design for safety assessment, *Information and Software Technology*, 73, 151-163.
- MacDonell, S.G. y Fletcher, T. (1998). Metric Selection for Effort Assessment in Multimedia Systems Development. Paper presented at the 5th International Sotware Metrics Symposium, Bethesda, MD, USA.
- Mandic, V. y Gvozdenovic, N. (2017). An extension of the GQM+ Strategies approach with formal causal reasoning. *Information and Software Technology* 88, 127–147.
- Manhart, P. y Schneider, K. (2004). Breaking the Ice for Agile Development of Embedded Software: An Industry Report. Paper presented at the Proc. of the 26th International Conference on Software Engineering (ICSE`04).
- Manlove, D. y Kan, S. H. (2007). Practical statistical process control for software metrics. *Software Quality Professional*, 9(4), 15.
- Matias, R., de Sena, G., Andrzejak, A. y Trivedi, K. (2016). Software Aging Detection Based on Differential Analysis: An Experimental Study. 2016 IEEE 27th International Symposium on Software Reliability Engineering Workshops
- Maxwell, K. (2002). Applied Statistics for Software Managers, Software Quality Institute Series: Prentice Hall. EE.UU.
- McCabe, T. (1976). Software Complexity Measure. *IEEE Transactions on Software Engineering*, 2, 308-320.
- McCabe, T. y Butler, W. (1989). Design complexity measurement and testing. *Communications of the ACM* 32(12), 1415-1425.
- Meidan, A., García-García, J. A., Ramos, I. y Escalona, M.J. (2018). Measuring Software Process: A Systematic Mapping Study. *ACM Comput. Surv.* 51(3), 58:1-58:32.
- Mellegård, N. (2017). Using weekly open defect reports as an indicator for software process efficiency. 27th International Workshop on Software Measurement and 12th International Conference on Software Process and

- Monden, A., Matsumura, T., Barker, M., Torii, K. y Basili, V. (2012). Customizing GQM models for software project monitoring (Article) IEICE Transactions on Information and Systems, E95(9), 2169-2182.
- Montini, D.A., Goncalves, C., da Silva, D.A., Vieira, L.A. y Marques, A. (2009). Using GQM for Testing Design Patterns in Real-Time and Embedded Systems on a Software Production Line. Paper presented at the ITNG, 2009, Information Technology: New Generations, Third International Conference on, Information Technology: New Generations.
- Morisio, M., Seaman, C.B., Parra, A.T., Basili, V., Kraft, S. E. y Condon, S.E. (2000). Investigating and improving a COTS-based software development. Paper presented at the Proceedings of the 22nd international conference on Software engineering ICSE '00.
- Munch, J.A., Fagerholm, F.A., Kettunen, P.A., Pagels, M.A. y Partanen, J.B. (2013). Experiences and insights from applying GQM+Strategies in a systems product development organisation Paper presented at the Proceedings - 39th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2013, Santander; Spain.
- Murdoch, J., Clark, G., Powell, A. y Caseley, P. (2003). Measuring safety: applying PSM to the system safety domain. Paper presented at the Proceedings of the 8th Australian workshop on Safety critical systems and software - Volume 33 SCS '03.
- NASA. (1981). Data Base Organization and User's Guide, National Aeronautics and Space Administration (NASA). Software Engineering Laboratory (SEL), EE.UU.
- Nelson, E. (1966). Management Handbook for the Estimation of Computer Programming Costs: Systems Development Corp.
- Nelson, L.S. (1999). Technical Aids-Notes on the Shewhart Control Chart. Journal of Quality Technology, 31(124-126).
- Nicolette, D. (2015). Software Development Metrics. . Shelter Island, NY., EEUU: Manning Publications, Co.
- Niessink, F. y Vliet, H.V. (2001). Measurement Programs Success factors revised. Information and Software Technology, 43, 617-628.
- Nuñez-Varela, A.S., Pérez-Gonzalez, H.G., Martínez-Perez, F.E. y Soubervielle-Montalvo, C. (2017). Source code metrics: A systematic mapping study, Journal of Systems and Software, 128, 164-197.
- Olsson, T. y Runeson, P. (2001). V-GQM: A Feed-Back Approach to Validation of a GQM Study. Paper presented at the Proc. of the Seventh International Software Metrics Symposium (METRICS'01).
- OMG (2018). Structured Metrics Metamodel (SMM), version 1.2. OMG. Object Management Group.
- Ott, L. (1996). The Early Days of Software Metrics. In A. Melton (Ed.), Software Measurement (7-25): International Thomson Computer Press.
- Pandian, C. R. (2004). Software Metrics – A Guide to Planning, Analysis, and Application: CRC Press.
- Pandian, C.R. (2003). Metrics are organization specifics. Software Metrics - A Guide to Planning, Analysis and Application, 19.
- Park, R. (1992). Software Size Measurement: A Framework for Counting Source Statements (CMU/SEI-92-TR-20). Pittsburgh, EE.UU., Software Engineering Institute, Carnegie Mellon University.
- Park, R., Goethert, W. y Florac, W.A. (1996). Goal-Driven Software Measurement - A Guidebook. Handbook CMU/SEI-96-HB-002: Software Engineering Institute.
- Parnas, D. (1975). The Influence of Software Structure on Reliability. Paper presented at the Proceedings of International Conference on Reliable Software.

- Paulk, M.C. (2001). Applying SPC to the Personal Software Process. Paper presented at the Proc. 10th Intl. Conf. Software Quality.
- Pavapootanont, S. y Prompoon, N. (2015). Defining usability quality metric for mobile game prototype using software attributes. 6th IEEE International Conference on Software Engineering and Service Sciences, nº 7339161, 730-736.
- Pérez, R., García, I., Ruiz, F., Polo, M. y Piattini, M. (2018). Mantenimiento y Evolución de Sistemas de Información. Ra-Ma, Madrid.
- Perry, D., Porte, A. y Votta, L. (2000). Empirical Studies of Software Engineering: A Roadmap. Future of Software Engineering: Anthony Finkelstein, ACM.
- Petersen, K.A , Gencel, C.B., Asghari, N.C. y Betz, S.D. (2014). An elicitation instrument for operationalising GQM+Strategies (GQM+S-EI) Empirical Software Engineering, 1-38.
- Pfleeger, S.L. (1997). Assessing Software Measurement. IEEE Software, 25-26.
- Piattini, M., Calvo Manzano, J., Cervera, J. y Fernández, L. (2003). Análisis y Diseño de Aplicaciones Informáticas de Gestión – Una perspectiva de Ingeniería del Software (2ª edición actualizada y ampliada ed.). Madrid: Ra-Ma.
- Poels, G. y Dedene, G. (2000). Distance-based software measurement: necessary and sufficient properties for software measures. Information and Software Technology, 42(1), 35-46.
- Prasad, R., Rao, B.S. y Kantam, R.R.L. (2011). Monitoring Software Reliability using Statistical Process control: An MMLE approach.
- Pressman, R. (2001). Ingeniería del Software. Un enfoque práctico (5th ed ed.): McGraw-Hill.
- Putnam, L.H. y Myers, W. (2003). Five Core Metrics: The Intelligence Behind Successful Software Management (U. S. Dorset House Publishing Co Inc. Ed.).
- Quesenberry, C. P. (1991). SPC Q charts for start-up processes and short or long runs. Journal of Quality Technology, 23(3), 213-224.
- Raczynski, B. y Curtis, B. (2008). Software data violate SPC's underlying assumptions. IEEE software, 25(3), 48-50.
- Radice, R. (2000). Statistical Process Control in Level 4 and 5 Organizations Worldwide. Paper presented at the Proceedings of the 12th Annual Software Technology Conference.
- Ram, P., Rodríguez, P. y Oivo, M. (2018). Software Process Measurement and Related Challenges in Agile Software Development: A Multiple Case Study. PROFES, 272-287.
- Rana, R. y Staron, M. (2015). Machine learning approach for quality assessment and prediction in large software organizations. (2015). 6th IEEE International Conference on Software Engineering and Service Science, artículo nº 7339243, 1098-1101.
- Reynoso, L., Genero, M. y Piattini, M. (2010). Refinement and Extension of SMDM, a Method for Defining Valid Measures. JUCS 16(21): 3210-3244.
- Rombach, H. D. (1990). Design measurement: some lessons learned. IEEE software, 7(3), 17-25.
- Rubey, R.J. y Hartwick, R.D. (1968). Quantitative Measurement Program Quality. Paper presented at the National Computer Conference.
- Russel, R. y Taylor, B. (2006). Operation Management. Cap. 4. 5^a Edición. : John Wiley & Sons.
- Sarcia', S.A. (2010). Is GQM+Strategies really applicable as is to non-software development domains? Paper presented at the In Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '10), New York, NY, USA.
- Sargut, K. U. y Demirörs, O. (2006). Utilization of statistical process control (SPC) in emergent software organizations: Pitfalls and suggestions. Software Quality Journal, 14(2), 135.

- Schneidewind, N. (1999). Can Metrics and Models be Applied Across Multiple Releases or Projects? Paper presented at the Proceedings of the Sixth International Metrics Symposium, Boca Raton, Florida.
- Scholtz, J. y Steves, M. P. (2004). A framework for real-world software system evaluations. Paper presented at the Proceedings of the 2004 ACM conference on Computer supported cooperative work CSCW '04.
- Selby, R.W. (2005). Measurement-Driven Dashboards Enable Leading Indicators for Requirements and Design of Large-Scale Systems. Paper presented at the Proceedings of the 11th IEEE International Symposium on Software Metrics (METRICS 2005).
- Serrano, M., Calero, C. y Piattini, M. (2002). Validating Metrics for Data Warehouses. Paper presented at the Proceedings of the Conference on Empirical Assessment in Software Engineering (EASE 2002), Keele, Reino Unido.
- Solingen, R.V. y Berghout, E. (1999). The Goal/Question/Metric Method, A Practical Guide for Quality Improvement of Software Development. London, England: McGraw-Hill International (UK).
- Solingen, R.V. y Berghout, E. (2001). Integrating Goal-Oriented Measurement in Industrial Software Engineering: Industrial Experiences with and Additions to the Goal/Question/Metric Method (GQM). Paper presented at the Proceedings Seventh International Software Metrics Symposium (METRICS'01).
- Solingen, R.V., Berghout, E. y Kooiman, E. (1997). Assessing Feedback of Measurement Data: Relating Schlumberger RPS practice to learning theory. Paper presented at the Proceedings of the 4th International Software Metrics Symposium (METRICS '97).
- Staron, M., Meding, W. y Caiman, M. (2013). Improving completeness of measurement systems for monitoring software development workflows. 5th International Conference on Software Quality Days, SWQD 2013, Lecture Notes in Business Information Processing, 133 LNBIP, 230-243
- Staron, M., Meding, W., Hansson, J., Höglund, C., Niesel, K. y Bergmann, V. (2014). Dashboards for Continuous Monitoring of Quality for Software Product under Development. En: Relating System Quality and Software Architecture, 209-229.
- Staron, M. Meding, W., Karlsson, G. y Nilsson, C. (2010). Developing measurement systems: An industrial case study. Journal of Software Maintenance and Evolution 23 (2), 89-107.
- Staron, M., Meding, W. y Nilsson, C. (2008). A framework for developing measurement systems and its industrial evaluation. Information and Software Technology 51 (4), 721-737.
- Staron, M. y Meding, W. (2015). Measurement-as-a-service - A new way of organizing measurement programs in large software development companies. 25th International Workshop on Software Measurement, IWSM 2015 and 10th International Conference on Software Process and Product Measurement. Lecture Notes in Business Information Processing 230, 144-159.
- Staron, M. y Meding, W. (2016). MeSRAM: A method for assessing robustness of measurement programs in large software development organizations and its industrial evaluation. Journal of Systems and Software 113, 76-100.
- Staron, M. y Meding, W. (2018). Software Development Measurement Programs - Development, Management and Evolution. Springer 2018, ISBN 978-3-319-91835-8.
- Staron, M. (2012). Critical role of measures in decision processes: Managerial and technical measures in the context of large software development organizations. Information and Software Technology 54 (8), 887-899.
- Tahir, T., Rasool, G. y Gencel, C. (2016). A systematic literature review on software measurement programs. Information and Software Technology 73, 101–121.
- Tahir, T., Rasool, G. y Noman, M. (2018). A systematic mapping study on software measurement programs in SMEs. E-informatica Software Engineering Journal 12 (1), 133-165. 73, 101–121.

- Tarhan, A. y Altunel, H. (2017). How do we measure the effects of agile transformation? Information needs and metrics for a medium-sized software company. 11th Turkish National Software Engineering Symposium, UYMS 2017; Alanya; Turkey; 18-20 October 2017, CEUR Workshop Proceedings 1980, 12-23 Kılıçaslan, F.N.aEmail Author,
- Tarhan, A. y Demirörs, O. (2006). Investigating suitability of software process and metrics for statistical process control. En: I. Richardson, & R. Messnarz (eds.), Software Process Improvement - Lecture Notes in Computer Science (Vol. 4257, pp. 88-99). Berlin/Heidelberg.
- Tarhan, A. y Yilmaz, S. (2014). Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process. *Information and Software Technology*, Volume 56, Issue 5, pp. 477-494.
- Trendowicz, A. (2013). Software Cost Estimation, Benchmarking, and Risk Assessment. Berlin, Springer-Verlag.
- Trinkenreich, B., Santos; G., Perini, M. y Conte, T. (2017). Eliciting Strategies for the GQM+Strategies Approach in IT Service Measurement Initiatives. 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, IEE Computer Society, 374-383.
- Trinkenreich, B., Santos, G. y Perini, M. (2018). SINIS: A GQM+Strategies-based approach for identifying goals, strategies and indicators for IT services. *Information and Software Technology* 100, 147–164.
- Tsunoda, M., Matsumura, T. y Matsumoto, K.-I. (2010). Modeling software project monitoring with stakeholders. 9th IEEE/ACIS International Conference on Computer and Information Science, ICIS 20102010, 723-728.
- Vianden, M., Lichter, H. y Steffens, A. (2013). Towards a maintainable federalist enterprise measurement infrastructure. Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement, IWSM-MENSURA 2013, 63-70.
- Vieira, M., Madeira, H., Cruz, S., Costa, M., y Cunha, J.C. (2011). Integrating GQM and Data Warehousing for the Definition of Software Ireland: Reuse MetricsLimerick.
- von Konsky, B.R., y Robey, M. (2005). A Case Study: GQM and TSP in a Software Engineering Capstone Project. Paper presented at the Proc. of the 18th Conference on Software Engineering Education & Training (CSEET'05).
- Weller, E., y Card, D. (2008). Applying SPC to software development where and why. *IEEE software*, 25(3), 48-50.
- Weller, E. F. (2000). Practical Applications of Statistical Process Control. *IEEE software*.
- Weyuker, E. (1988). Evaluating software complexity measures. *IEEE Transactions on Software Engineering*, 14(9), 1357-1365.
- Wheeler, D.J., y Chambers, D.S. (1992). Understanding Statistical Process Control (2^aed.), SPC Press.
- Whitmire, S. (1997). Object Oriented Design Measurement: John Wiley & Sons, Inc.
- Wiegers, K.E. (1999). A software metrics primer. *Software Developer Magazine*, 5(7).
- Wohlin, C., Runeson, P., Höst, M., Ohlson, M., Regnell, B. y Wesslén, A. (2000). *Experimentation in Software Engineering: An Introduction*: Kluwer Academic Publishers.
- Wolverton, R.W. (1974). The Cost of Developing Large-Scale Software. *IEEE Transactions on Computer*, 23(6), 615-636.
- Yahaya, J.H., Abidin, Z.N.Z., Ali, N.M. y Deraman, A. (2013). Software ageing measurement and classification using Goal Question Metric (GQM) approach. Proceedings of 2013 Science and Information Conference, SAI 2013, London, R.U..
- Yahya, F., Walters, R. J. y Wills, G. B. (2017). Using Goal-Question-Metric (GQM) Approach to Assess Security in Cloud Storage. In V. Chang, M. Ramachandran, R. J. Walters & G. Wills (Eds.), *Enterprise Security: Second International Workshop, ES 2015, Vancouver, BC, Canada*, pp. 223-240. Springer International Publishing.

Yamada, S. (2016). Statistical Process Control for OSS Projects and Optimal Release Policies. 5th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), Sep. 7-9, 2016, AIIT, Amity University Uttar Pradesh, Noida, India.

Yin, B. H. y Winchester, J. (1978). The establishment and use of measures to evaluate the quality of software design. Paper presented at the Proceedings of the Software Quality Assurance Workshop.

Zubrow, D. (1998). Measurement With a Focus: Goal-Driven Software Measurement? CrossTalk, 11(9), 24-26.

Zuse, H. (1991). Software Complexity Measures and Methods. Berlin: Walter de Gruyter.

Zuse, H. (1998). A Framework of Software Measurement. Berlin: Walter de Gruyter.



Dogram Code

Accede a la Biblioteca Online +300 libros en PDF Gratis!

<https://dogramcode.com/biblioteca>

Únete al Canal de Telegram

https://t.me/bibliotecagratis_dogramcode

Únete al grupo de Facebook

<https://www.facebook.com/groups/librosyrecursosdeprogramacion>

<https://dogramcode.com/programacion>