aemes TI
Asociación Española para la Gobernanza, la Gestión y la Medición de las Tecnologías de la Información

Revista de Procesos y Métricas

1 de enero

2010

De las Tecnologías de la Información

VOLUMEN 7, NÚMERO 1 ENERO-ABRIL 2010
ISSN 1698-2029

Revista de Procesos y Métricas

De las Tecnologías de la Información

Volumen 7 Número 1

Revista fundada por la Asociación Española para la Gobernanza, la Gestión y la Medición de las Tecnologías de la Información (AEMES) <http://www.aemes.org>

ISSN: 1698-2029. Nº Depósito: M23879-2006

Revista de Procesos y Métricas

De las Tecnologías de la Información

Carta del Presidente de AEMES

Estimado Socio,

Comienza una nueva etapa en el desarrollo de AEMES, una etapa en la que el principal objetivo va a ser que la asociación se convierta en referente y punto de encuentro de los profesionales con dedicación o interés en la gobernanza, la gestión y la medición de las tecnologías de la información.
Esta nueva etapa no solo se va a ver representada por el cambio del logotipo y de la denominación de la asociación, si no por la potenciación de la comunicación entre los asociados y por el acercamiento de las actividades de éstos al resto de la sociedad.
La renovada revista de AEMES pretende convertirse en un vehículo de información de calidad para todos aquellos interesados en las iniciativas promovidas por la asociación (la Gobernanza de TI, la Mejora de Procesos, las Métricas e Indicadores y la Formación).
Desde la Junta Directiva de la AEMES invitamos a asociados y visitantes a participar en las actividades de la asociación y a colaborar en la revista.

José Domingo Carrillo Verdún
Presidente de AEMES

Carta del Editor Jefe de RPM

Estimado Lector,
tiene en sus manos un nuevo número de RPM, que continúa una serie de volúmenes que se extienden durante seis años. Es un largo camino para una revista que, desde su fundación, ha sido uno de los activos más importantes de nuestra asociación, por el gran interés que ha despertado en sus lectores.
El año pasado, coincidiendo con el quinto aniversario de la revista, desde la junta directiva de AEMES se decidió que era el momento de iniciar una nueva etapa en la revista que, manteniendo sus orígenes, supusiese una clara renovación y actualización que permitiese, tanto aumentar el interés de sus lectores, como prepararla para el futuro
El ejemplar que tiene en sus manos es el primero de esta nueva serie. Como podrá observar se han introducido un gran número de cambios y mejoras. El diseño se ha renovado completamente, haciéndolo más atractivo y fácil de leer. Se han introducido un gran número de secciones nuevas, empezando por una nueva sección de artículos de divulgación, en los que, las empresas asociadas irán presentando estudios sobre aspectos de actualidad. Una nueva de nuevos libros de interés o Conferencias relacionadas con los aspectos de interés de AEMES hacen que la revista sea ahora no sólo más interesante sino más útil.
Desde aquí me gustaría hacer un llamamiento a todos los asociados a participar con colaboraciones o ideas que permitan hacer nuestra revista cada día mejor.
Reciban un cordial saludo,

Dr. Juan J. Cuadrado-Gallego
Editor Jefe de RPM

Revista de Procesos y Métricas

De las Tecnologías de la Información

Índice

Volumen 7 Número 1 Enero-Abril 2010

Gobernanza de TI > Definición

Cesar Nistal
Atos Origin, Atos Consulting
www.atosorigin.com

Como punto de partida quizás sea interesante ofrecer una definición -posiblemente, una más- del concepto de gobierno de TI: *"El Buen Gobierno Corporativo de la Información y sus Tecnologías afines es una responsabilidad de los Consejos de Administración de las organizaciones, u órganos equivalentes, y de sus equipos de Alta Dirección; es una parte integrante del Buen Gobierno Corporativo; es un sistema, en definitiva, mediante el cual se dirige y controla el uso, presente y futuro, de las TI dentro de dichas organizaciones. Y todo ello, aun cuando la Dirección de Informática no esté representada en los citados órganos directivos".*

En el contexto fijado por la anterior definición, se presumirá, asimismo, un gobierno de TI constituido por una serie de mecanismos, entre los que cabrá identificar:

- Conductas éticas, liderazgo, compromiso y respaldo.
- Estructuras organizativas, con papeles, responsabilidades e interrelaciones bien definidos, para la toma de decisiones sobre las inversiones en actividades de la organización soportadas por las TI.
- Procesos naturalmente agrupados, y generalmente aceptados, que cubran el Ciclo de Vida completo de las TI dentro de la organización.
- Actividades de supervisión, medición y comunicación.

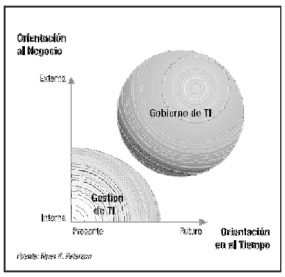
Asimismo, ha de apuntarse que cualquier iniciativa que persiga materializar los anteriores mecanismos de Buen Gobierno Corporativo de las TI en el terreno de la práctica, dentro de una organización, estará, en realidad, orientada a Ayudar a la organización a alcanzar sus objetivos estratégicos mediante la aportación de valor por parte de las TI, al tiempo que se mitigan los riesgos que pueden afectar a la propia organización, derivados del empleo de dichas tecnologías, y a asegurar un adecuado comportamiento en el uso presente y futuro de aquellas.

Buen Gobierno frente a Gestión

Por lo dicho hasta ahora, podría llegarse a una primera interpretación básica que pasaría por definir el buen gobierno de TI como el proceso de toma de decisiones en torno a las tecnologías de la información. Sin embargo, ello lleva a la siguiente reflexión: en la toma de toda decisión intervienen, al menos, tres factores; a saber:

- ¿Quién toma la decisión (el Consejo, el Consejero Delegado, el Director de TI...)?
- ¿Cómo se toma la decisión (de manera individual, colegiada, unilateral, consensuada...)?
- ¿Qué decisión se toma?

El buen gobierno de TI guarda más relación con el ¿Quién? y con el ¿Cómo? (estructuras organizativas para la toma de decisiones); mientras que la cuestión del ¿Qué? -y, particularmente, la ejecución de ese "qué"- queda más en el terreno de la gestión del día a día de las TI. De este modo queda patente la idea de que el gobierno de TI tiene una orientación más centrada en su contribución al negocio y en su posicionamiento para ayudar a aquél a alcanzar retos futuros; al tiempo que la gestión de TI, con una perspectiva más cortoplacista, se fija en la eficacia y eficiencia de los servicios, los productos y las operaciones.



Un mensaje, también, para la Dirección de Informática

La adopción de una cierta ortodoxia en la definición del concepto de Buen Gobierno Corporativo de las TI, no debe hacer olvidar el carácter fronterizo de esta nueva disciplina.

Como se ha apuntado, el gobierno de las TI, en sentido estricto, no es sino una responsabilidad exclusiva de la Alta Dirección de la organización, mediante la que habrá de buscarse, por un lado, la transparencia que el mercado y los diferentes grupos de interés demandan y, por otro, la garantía de conformidad con las diferentes normativas, internas y/o externas, a que pudiera estar expuesta la entidad. Un razonamiento tal, sin duda, no resulta del todo erróneo, por cuanto en él se encierra el origen del concepto de "Buen Gobierno Corporativo", que indudablemente va ligado a la necesidad de que las personas al frente de las organizaciones "hagan las cosas correctas, del modo correcto", esto es, obren bien, cumpliendo con las prescripciones impuestas y mitigando los riesgos que, derivados de su conducta, puedan afectar a la organización.

No obstante, si bien resulta correcto ese planteamiento, se hace insuficiente si se considera que, también, para el área de TI hay una oportunidad en la adopción de un marco de buen gobierno corporativo. Se trata de la oportunidad de ganar o, en algunos casos, recuperar, un mayor grado de visibilidad dentro de la propia organización (mayor visibilidad desde la perspectiva del resto de áreas de la entidad). En este caso, esa ganancia en visibilidad del área de TI habrá de venir dada por la mayor aportación de éstas al negocio, materializada en un mayor rendimiento, y por una correcta medida y mejor comunicación ("venta interna") de dicha aportación.

De todo ello se concluye que el buen gobierno de TI es una competencia compartida entre la Alta Dirección de las organizaciones y la función de TI, cuyos beneficios se muestran en una doble vertiente: ofreciendo una mayor transparencia y una ganancia en visibilidad.
En el caso de esta última, puede venir, además, acompañada de una elevación de la función de TI hacia posiciones más cercanas al negocio, como recoge el lema elegido para este documento.

César Nistal, Atos Origin, Atos Consulting

César Nistal es PMP, Auditor ISO 20000. Es un profesional con más de 15 años de experiencia en TI que actualmente desarrolla su actividad como Gerente de Consultoría de Negocio en Atos Consulting. Ha colaborado con clientes de diversos sectores (banca, seguros, industria, telecomunicaciones, sanidad, defensa,…) en proyectos relacionados con la gobernanza de TI y ha formado parte de los equipos de certificación de Atos Origin Castilla y León (CMMI nivel 3), Atos Origin LSIS (CMMI nivel 3), Atos Origin - Major Events (CMMI nivel 3) y Atos Origin LSMO (ISO 20000). César Nistal es Vocal de la Junta Directiva de AEMES y, actualmente, es responsable del sitio web de la asociación.

Atos Origin es una compañía internacional de servicios de tecnologías de la información. Su objetivo es transformar la visión estratégica de sus clientes en resultados mediante una mejor utilización de soluciones de Consultoría, Integración de Sistemas y Outsourcing. La compañía emplea 50.000 profesionales en 40 países, y su facturación anual es de 5.500 millones de euros. Atos Origin es partner tecnológico mundial para los Juegos Olímpicos, y sus clientes son grandes compañías internacionales de todos los sectores de actividad. Atos Origin cotiza en el mercado Eurolist de París y ejerce sus actividades con los nombres Atos Origin, Atos Worldline y Atos Consulting. Atos Consulting, la práctica internacional de consultoría de Atos Origin, es un proveedor líder de servicios de consultoría de negocio, procesos y tecnología. Con más de 2.500 empleados en todo el mundo, su actividad se centra en la entrega de soluciones probadas y pragmáticas para los mercados de Telecomunicaciones, Industria, Servicios Financieros y Sector Público.

Artículos de Investigación
(con revisión por pares)

Umbrales relativos. Caso de estudio para incorporar métricas en la detección de *Bad Smells*

Raúl Marticorena[1] , Carlos López[1], Yania Crespo[2], Esperanza Manso[2]
[1] Universidad de Burgos
{rmartico, clopezno }@ubu.es
[2] Universidad de Valladolid
{yania,manso}@infor.uva.es

Abstract: *To detect flaws, bad smells, etc, we often use quantitative methods: metrics or measures. It is common in practice to use thresholds setting the correctness of the measures. Most of the current tools use absolute values. Nevertheless, there is a certain concern about threshold applications on obtained values. Current work tries to accomplish case studies about thresholds on several products and different versions. By other side, product domain and size could also affect the results. We tackle if it is correct to use absolute vs. relative thresholds, seeing that effects could have in metric collection and bad smell detection.*

Resumen: *Para la detección de defectos, errores, etc. se utilizan en muchas ocasiones métodos cuantitativos: métricas. El empleo de umbrales (thresholds) para determinar la corrección de los valores obtenidos está ampliamente difundido. Sin embargo, existe una cierta discusión sobre la aplicación de dichos umbrales a los valores obtenidos. Este trabajo pretende realizar varios casos de estudio sobre la posible aplicación de umbrales sobre distintos productos. El dominio y tamaño del producto medido pueden afectar sobre las consideraciones a tomar. Este trabajo trata de establecer si es correcto utilizar umbrales absolutos frente a relativos, así como el efecto que puede tener sobre soluciones de recuperación de métricas y detección* de *bad smells sobre el código.*

Palabras Clave: *Métricas de Código, Umbrales, Proceso de Medición, Herramientas de Medición, Bad Smells,*

1. Contexto inicial

En anteriores trabajos [3,11], se ha planteado el uso de *frameworks* con el fin de dar un soporte completo al proceso de *refactoring* [4]: extracción de información del códigos fuente, recolección de métricas para la detección de *bad smells*, soporte de refactorizaciones y recuperación del código transformado.

Sin embargo en dicho proceso quedan puntos por cubrir. En particular, la relación entre métricas y *bad smell* definida en base a umbrales absolutos, en anteriores soluciones [3]. Aunque dichos umbrales podían ser personalizados, ésta era una labor para el gestor del producto. Quedan varias cuestiones que se plantearon a la hora de utilizar dicho enfoque sobre el código fuente: ¿qué valores utilizar?, ¿son adecuados dichos valores?, ¿son correctos para el producto concreto que utilizamos?

Este trabajo pretende contestar estas preguntas, utilizando un caso de estudio con varios productos de tamaño medio y centrándose también en la evolución de alguno de ellos. Las conclusiones obtenidas deberían dejar claro la necesidad de utilizar valores relativos.

El resto del artículo está organizado de la siguiente forma, en la sección 2 se muestra un conjunto de trabajos relacionados, la sección 3 desarrolla el caso de estudio centrándose en varios productos y sus versiones. En la sección 4 se establece la aplicación de los resultados para la detección de *bad smells*. Por último, en la sección 5 se muestran unas conclusiones de la solución propuesta.

2. Trabajos relacionados

En la mayoría de entornos que incluyen la recolección de métricas se empieza a incluir la posibilidad de fijar límites (umbrales) a los valores recogidos por las métricas. Generalmente, es el propio programador, normalmente bajo las guías de desarrollo de su empresa, el que establece estos valores. Pero estos filtros se establecen para todos los productos y los resultados obtenidos no sugieren siempre defectos catalogados.

Existen trabajos en la línea de detección de defectos del diseño. En los trabajos de Marinescu [9,10] se propone lo que denomina estrategias de diseño (*design strategies*). Dichas estrategias se definen en base a métricas y aplicadas sobre la información de un metamodelo. El metamodelo recoge información del código, pero está pensado básicamente para consultas (todas las operaciones se traducen a sentencias SQL sobre las tablas) y no para completar el proceso de refactorización del código. En dicho trabajo, se plantea la utilización de filtros absolutos y relativos, pero no se menciona ningún estudio para tomar en cuenta la idoneidad de unos u otros.

En los trabajos de Mäntyla [7,8] se catalogan los defectos denominados *bad smell*, intentando ligarlos a un juego de métricas. Esta última asignación se sugiere, pero no se profundiza en exceso sobre la problemática de los umbrales, seleccionando en su mayoría umbrales absolutos. Mäntyla también muestra a través de validaciones con expertos el problema que detectar bad smell está sujeto a decisiones de la intuición humana.

Por otro lado, en [12], Tourwé y Mens proponen la detección de oportunidades de refactorización utilizando consultas en un entorno de meta-programación lógica. Ellos definen consultas para llevar acciones de corrección en el sistema. Como continuación de este trabajo, en [1], Muñoz usa un conjunto de preguntas lógicas basadas en métricas orientadas a objetos para detectar estos *bad smells* con umbrales absolutos.

Los umbrales también se han tratado por French en [5]. En el trabajo se propone un sistema basado en métodos estadísticos para determinar umbrales de diferentes productos, éstos no se aplican ni en la detección de *bad smells* ni para comprobar los efectos sobre varias versiones del mismo producto.

Partiendo de estos trabajos previos se pretende resolver ciertas cuestiones:

- Comprobar la corrección de la utilización de umbrales absolutos frente a relativos para una futura detección de defectos código o bad smells.
- Influencia del tipo y tamaño del producto software: frameworks vs. bibliotecas.
- Adecuación de la solución en diferentes versiones del mismo producto.

3. Caso de estudio

Partiendo de un conjunto de productos se inicia la recolección y estudio de sus métricas obtenidas para establecer qué hipótesis son correctas respecto al uso de umbrales.

3.1 Primera fase: Comparación entre productos

Se ha realizado un estudio comparativo entre seis productos. Se han tomado diferentes productos, en su mayoría con versiones estables utilizadas durante un periodo mayor de un año, y de tamaños medios a grandes, obviando el uso de los denominados "toys" o casos de pequeño tamaño, casi sin funcionalidad.

Los productos elegidos son:

- jfreechart-1.0.0.pre2 (629 clases)
- jhotdraw-6.0b1 (496 clases)
- struts-1.2.8 (273 clases)
- jcoverage-1.0.5 (90 clases)
- easymock-1.0.5 (47 clases)
- junit-3.8.1 (46 clases)

Todos estos productos están escritos en un lenguaje orientado a objetos como Java, puesto que los resultados extraídos se quieren aplicar sobre trabajos previos en dicho paradigma. En el estudio, se ha utilizado Eclipse y su plugin Metrics-1.3.6 como herramienta de recolección de métricas. Esto condiciona a la utilización de productos escritos en Java, aunque a juicio de los autores, se cree que el método es generalizable a otros lenguajes orientados a objetos.

Las métricas seleccionadas han sido tomadas sólo sobre clases, seleccionando métricas que muestren valores de: tamaño, complejidad, cohesión, herencia y especialización [2,6].

- NOF número de atributos
- NOM número de métodos
- WMC complejidad ciclomática
- LCOM ausencia de cohesión
- DIT profundidad en el árbol de herencia
- NSC número de hijos
- SIX coeficiente de especialización
- NORM número de métodos redefinidos

Para cada uno de estas métricas se han recogido los valores y calculado: media, media acotada (eliminando el 15% de valores extremos), desviación estándar, cuartil Q1, mediana, cuartil Q3, mínimo y máximo.

3.2 Conclusiones parciales

De los resultados obtenidos, ver Fig.1, se pueden deducir las siguientes conclusiones:

- Las distribuciones no son simétricas, las diferencias obtenidas entre las medias y medianas, así como la proximidad de la mediana al cuartil Q3 lo corroboran. En la mayoría de casos nos encontramos con distribuciones con asimetría positiva (cola de la distribución a la derecha).
- Las diferencias entre valores mínimos y máximos es muy grande, y además en cada producto es muy distinto. Esto sugiere datos muy dispersos, y con intervalos muy diferentes entre los productos.
- El tamaño del producto (número de clases) está ligeramente correlacionado con ciertas métricas lo que sugiere que el tamaño podría influir en los umbrales. Esto es más acusado en métricas de tamaño como NOF, NOM y WMC, con alta correlación entre ellas. Por el contrario métricas como LCOM, DIT sufren pocas variaciones entre los distintos productos.

Como ejemplo se muestra en la figura 2 un diagrama de cajas (eliminando mínimos y máximos), centrándonos en los cuartiles Q1 y Q3 como límites de las cajas. Los productos se ordenan de izquierda a derecha, de mayor a menor número de clases. Como se puede observar gráficamente las diferencias de distribución entre los productos son apreciables.

En trabajos previos, establecimos la posibilidad de utilizar umbrales de métricas con el objetivo de detectar defectos (en diseño no en funcionalidad). Como primera aproximación, se establecieron esos umbrales en base a valores absolutos. Los resultados de este estudio empírico indican que los valores deberían estar ajustados a los productos concretos.
Otro factor que podría tener influencia en los resultados es la clase de producto (observar el tipo de relleno en de las cajas en la figura 2). Productos similares como: frameworks de pruebas (junit y easymock), frameworks de desarrollo (struts y jhotdraw) y bibliotecas (jcoverage y jfreechart), presentan grandes diferencias entre los valores mínimo y máximo.

A la vista de estos resultados, aparece una nueva hipótesis: *la ausencia de umbrales absolutos genera un cierta degeneración de las valores de las métricas (cuando los productos incrementan su tamaño las métricas están descontroladas)*. Para comprobar esta hipótesis, se presenta un nuevo caso de estudio con diferentes versiones de algunos productos.

3.3 Segunda fase: Evolución de versiones

Se han tomado diferentes versiones de tres de estos productos: jfreechart, jhotdraw y junit. Se muestran las versiones y número de clases de las versiones. Las versiones muestran la evolución del producto en un periodo medio de tiempo:

	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean JFreeChart 1.0.0-pre2	2,40	10,08	22,98	0,21	2,55	0,36	0,16	0,69
Bounded mean (15%)	1,41	7,45	15,87	0,17	2,47	0,04	0,08	0,46
Q3	3,00	11,00	25,00	0,50	3,00	0,00	0,14	1,00
Q2 Median	1,00	5,00	9,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	3,00	6,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	5,05	15,01	38,82	0,32	1,14	1,48	0,37	1,23
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	48,00	166,00	490,00	1,00	7,00	16,00	3,20	9,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean Junit-3.8.1	2,17	8,13	15,70	0,21	2,70	0,28	0,18	0,35
Bounded mean (15%)	1,50	6,53	12,33	0,18	2,58	0,15	0,09	0,28
Q3	2,00	9,75	15,75	0,50	3,75	0,00	0,12	1,00
Q2 Median	1,00	4,50	8,00	0,00	2,00	0,00	0,00	0,00
Q1	0,00	2,00	4,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	3,59	10,35	20,42	0,33	1,84	0,72	0,45	0,60
Minimum	0,00	0,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	18,00	62,00	106,00	0,91	6,00	3,00	2,00	3,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean Jcoverage-1.0.5	1,49	4,35	9,56	0,24	1,78	0,39	0,81	0,28
Bounded mean (15%)	1,23	3,70	8,17	0,20	1,62	0,19	0,10	0,21
Q3	2,00	5,00	14,00	0,50	2,00	0,00	0,00	0,00
Q2 Median	1,00	3,00	5,00	0,00	1,00	0,00	0,00	0,00
Q1	0,00	2,00	3,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	1,87	4,46	9,59	0,34	1,05	0,96	0,37	0,52
Minimum	0,00	0,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	7,00	25,00	46,00	1,00	5,00	4,00	1,67	2,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean easymock-2.0	1,41	5,83	12,54	0,15	1,24	0,09	0,12	0,33
Bounded mean (15%)	1,24	4,13	8,32	0,11	1,08	0,00	0,02	0,16
Q3	2,00	5,00	13,50	0,33	1,00	0,00	0,00	0,00
Q2 Median	1,00	3,00	3,50	0,00	1,00	0,00	0,00	0,00
Q1	1,00	3,00	3,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	1,34	7,51	19,25	0,24	0,67	0,46	0,41	0,73
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	6,00	38,00	105,00	0,85	4,00	3,00	2,00	3,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean struts-1.2.8	2,91	8,60	18,84	0,28	2,59	0,46	0,51	0,96
Bounded mean (15%)	2,09	6,66	13,21	0,25	2,45	0,24	0,33	0,67
Q3	4,00	11,00	22,00	0,67	4,00	1,00	0,60	1,00
Q2 Median	2,00	4,00	8,00	0,00	2,00	0,00	0,00	0,00
Q1	0,00	2,00	3,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	4,56	11,02	29,13	0,36	1,48	1,13	0,95	2,04
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	40,00	82,00	260,00	0,98	7,00	10,00	5,00	28,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean JHotDraw60b1	1,40	9,51	13,36	0,16	2,84	0,57	0,31	0,73
Bounded mean (15%)	1,09	7,72	10,31	0,11	2,68	0,07	0,16	0,38
Q3	2,00	11,00	14,00	0,00	4,00	0,00	0,32	1,00
Q2 Median	1,00	7,00	9,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	4,00	5,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	1,86	10,40	16,76	0,30	1,49	3,84	0,74	1,70
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	19,00	90,00	158,00	1,50	9,00	71,00	8,00	19,00

Figura 1 Resultados globales

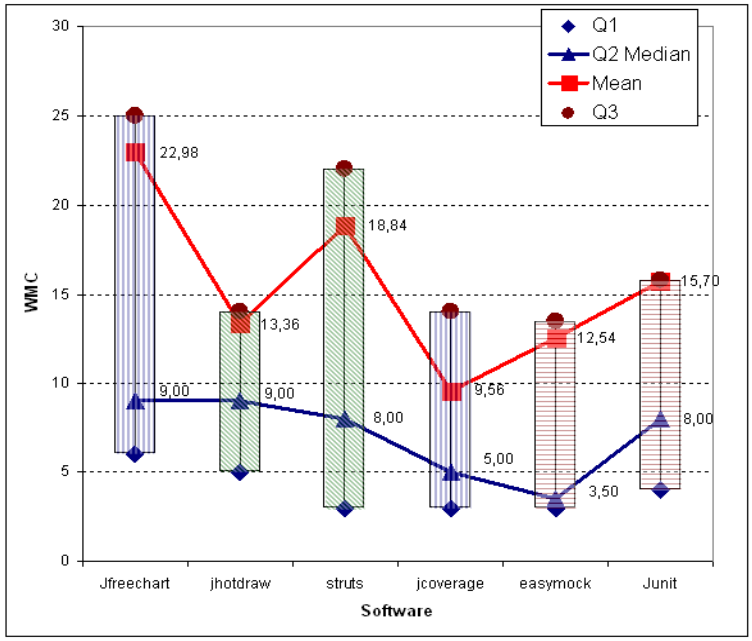


Figura 2 Distribución de la métrica WMC

- jfreechart-1.0.1 (691 clases, 2006-01-27)
- jfreechart-1.0.0-pre2 (629 clases, 2005-03-10)
- jfreechart-0.9.21 (570 clases, 2004-09-10)
- jfreechart-0.9.7 (492 clases, 2003-04-17)
- jfreechart-0.9.4 (326 clases, 2002-10-18)

	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jfreechart-1.0.1	2,22	9,94	22,42	0,19	2,53	0,33	0,16	0,69
Bounded mean (15%)	1,27	7,27	15,25	0,15	2,46	0,03	0,09	0,48
Q3	2,00	11,00	23,00	0,40	3,00	0,00	0,17	1,00
Q2 Median	1,00	5,00	9,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	4,00	7,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	4,86	15,18	39,39	0,31	1,12	1,41	0,35	1,18
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	46,00	173,00	513,00	1,00	7,00	14,00	3,33	8,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jfreeChart-1.0.0-pre2	2,40	10,08	22,98	0,21	2,55	0,36	0,16	0,69
Bounded mean (15%)	1,41	7,45	15,87	0,17	2,47	0,04	0,08	0,46
Q3	3,00	11,00	25,00	0,50	3,00	0,00	0,14	1,00
Q2 Median	1,00	5,00	9,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	3,00	6,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	5,05	15,01	38,82	0,32	1,14	1,48	0,37	1,23
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	48,00	166,00	490,00	1,00	7,00	16,00	3,20	9,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jfreechart-0.9.21	2,38	9,99	22,47	0,21	2,52	0,36	0,16	0,66
Bounded mean (15%)	1,41	7,33	15,44	0,17	2,45	0,05	0,08	0,44
Q3	2,00	12,75	26,00	0,50	3,00	0,00	0,16	1,00
Q2 Median	1,00	5,00	9,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	3,00	6,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	4,93	15,20	38,66	0,32	1,12	1,47	0,37	1,20
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	47,00	155,00	473,00	0,96	7,00	16,00	3,00	8,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jfreechart-0.9.7	2,14	7,03	15,63	0,20	3,21	0,31	0,17	0,49
Bounded mean (15%)	1,22	5,06	11,08	0,15	3,07	0,04	0,07	0,28
Q3	2,00	9,00	19,00	0,50	4,00	0,00	0,09	1,00
Q2 Median	0,00	3,00	6,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	1,00	3,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	4,55	10,65	24,45	0,32	1,97	1,34	0,44	1,02
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	39,00	87,00	203,00	1,00	7,00	15,00	3,00	7,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jfreechart-0.9.4	2,69	7,77	18,00	0,26	3,02	0,40	0,27	0,61
Bounded mean (15%)	1,71	6,13	13,51	0,22	2,85	0,08	0,11	0,32
Q3	3,00	11,00	24,00	0,62	4,00	0,00	0,16	1,00
Q2 Median	1,00	4,00	8,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	1,00	3,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	4,87	9,70	25,11	0,35	1,95	1,49	0,70	1,34
Minimum	0,00	0,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	39,00	60,00	195,00	1,00	7,00	16,00	6,00	8,00

Figura 3 Resultados JFreeChart

En el caso de jhotdraw las versiones, número de clases y fechas son:

- jhotdraw-6.0b1 (497 clases, 2004-02-01)
- jhotdraw-5.4b2 (478 clases, 2004-01-31)
- jhotdraw-5.3 (208 clases, 2002-01-20)
- jhotdraw-5.2 (149 clases, 2001-02-18)

	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean JHotDraw60b1	1,40	9,51	13,36	0,16	2,84	0,57	0,31	0,73
Bounded mean (15%)	1,09	7,72	10,31	0,11	2,68	0,07	0,16	0,38
Q3	2,00	11,00	14,00	0,00	4,00	0,00	0,32	1,00
Q2 Median	1,00	7,00	9,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	4,00	5,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	1,86	10,40	16,76	0,30	1,49	3,84	0,74	1,70
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	19,00	90,00	158,00	1,50	9,00	71,00	8,00	19,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jhotdraw54b1	1,41	9,67	13,89	0,16	2,90	0,58	0,32	0,73
Bounded mean (15%)	1,12	7,85	10,80	0,11	2,75	0,08	0,17	0,39
Q3	2,00	11,00	15,00	0,00	4,00	0,00	0,33	1,00
Q2 Median	1,00	7,00	9,00	0,00	3,00	0,00	0,00	0,00
Q1	1,00	4,00	6,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	1,81	10,33	16,88	0,30	1,48	3,89	0,75	1,71
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	16,00	88,00	148,00	1,50	9,00	71,00	8,00	19,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jhotdraw53	1,83	9,12	15,51	0,27	2,65	0,86	0,51	1,21
Bounded mean (15%)	1,46	7,07	11,60	0,23	2,43	0,20	0,41	0,97
Q3	3,00	10,00	18,00	0,63	3,00	0,00	0,75	2,00
Q2 Median	1,50	6,50	12,50	0,00	1,00	0,00	0,00	0,00
Q1	0,00	3,00	4,75	0,00	2,00	0,00	0,00	0,00
Standard Deviation	2,38	10,87	20,54	0,35	1,66	3,53	0,66	1,66
Minimum	0,00	0,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	17,00	72,00	146,00	1,50	8,00	40,00	3,00	12,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jhotdraw52	1,83	8,30	13,53	0,26	2,81	0,68	0,56	1,28
Bounded mean (15%)	1,52	6,37	10,25	0,23	2,60	0,24	0,48	1,06
Q3	3,00	10,00	15,25	0,60	3,00	0,00	1,00	2,00
Q2 Median	1,00	5,00	8,00	0,00	2,00	0,00	0,28	1,00
Q1	0,00	3,00	4,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	2,19	9,82	16,84	0,33	1,70	1,91	0,66	1,69
Minimum	0,00	0,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	14,00	61,00	108,00	1,50	8,00	12,00	3,11	12,00

Figura 4 Resultados JHotDraw

En el caso de junit las versiones y número de clases son:

- junit-3.8.1 (47 clases)
- junit-3.2 (32 clases)
- junit-2.1 (19 clases)

	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean Junit 3.8.1	2,17	8,13	15,70	0,21	2,70	0,28	0,18	0,35
Bounded mean (15%)	1,50	6,53	12,33	0,18	2,58	0,15	0,09	0,28
Q3	2,00	9,75	15,75	0,50	3,75	0,00	0,12	1,00
Q2 Median	1,00	4,50	8,00	0,00	2,00	0,00	0,00	0,00
Q1	0,00	2,00	4,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	3,59	10,35	20,42	0,33	1,84	0,72	0,45	0,60
Minimum	0,00	0,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	18,00	62,00	106,00	0,91	6,00	3,00	2,00	3,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean Junit 3.2	2,72	7,94	14,75	0,25	2,56	0,19	0,13	0,34
Bounded mean (15%)	1,68	5,96	11,29	0,22	2,43	0,04	0,09	0,25
Q3	3,00	11,00	18,50	0,50	3,50	0,00	0,19	1,00
Q2 Median	1,00	3,50	5,50	0,00	2,00	0,00	0,00	0,00
Q1	0,00	2,00	2,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	4,85	11,65	21,05	0,34	1,93	0,64	0,24	0,65
Minimum	0,00	0,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	20,00	60,00	103,00	0,92	6,00	3,00	0,75	3,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean Junit 2.1	2,16	8,11	14,05	0,22	2,53	0,32	0,31	0,58
Bounded mean (15%)	1,35	7,18	12,41	0,19	2,41	0,18	0,17	0,47
Q3	2,00	8,50	18,00	0,50	3,00	0,00	0,26	1,00
Q2 Median	1,00	4,00	6,00	0,00	2,00	0,00	0,00	0,00
Q1	0,00	4,00	4,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	4,06	8,52	15,30	0,31	1,61	0,82	0,70	0,84
Minimum	0,00	1,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	18,00	31,00	55,00	0,89	6,00	3,00	3,00	3,00

Figura 5 Resultados JUnit

Para cada uno de estos productos se han recogido las métricas anteriormente mencionadas. Se calculan de nuevo: media, media acotada, desviación estándar, cuartil Q1, mediana, cuartil Q3, mínimo y máximo.

Como se puede observar (ver Figuras 3-5), los resultados para cada producto se mantienen entre versiones, con pequeñas variaciones. Esto sugiere que una vez que el producto es estable, pese a que su tamaño aumenta (llegando en casi todos los casos estudiados a duplicarse el número de clases), los umbrales establecidos pueden ser relativamente correctos.

3.4 Conclusiones del estudio

De los resultados obtenidos se pueden deducir las siguientes conclusiones:

- Los umbrales se deben establecer de manera relativa al producto.
- Estos umbrales se pueden mantener entre distintas versiones del mismo producto.
- El tipo de producto (*framework* / biblioteca) no determina el cómo deben ser estos umbrales.

A partir de estas conclusiones se plantean nuevos problemas. Ante productos nuevos sería difícil decidirse por unos umbrales iniciales. Como primera estimación se podrían tomar valores de productos similares (mismo dominio, funcionalidad similar, tamaño estimado). En el caso de disponer de varias versiones del producto, el tomar los valores de dichas versiones parece una buena medida inicial. En ambos casos posiblemente sea necesario realizar algún ajuste.

4. Aplicación de umbrales relativos

En anteriores trabajos, se ha mostrado la utilidad del uso de métricas como indicador de la presencia de lo que se denominan “malos olores” o síntomas (*bad smells*). Este vocabulario, es propio de refactoring, aunque se puede generalizar a defectos del software. Se ha planteado el uso de umbrales para sugerir la presencia de dichos defectos. Por otro lado, se definió un framework que soportase todos estos conceptos. Sin embargo la definición de los umbrales es algo que quedó abierto [5]. En la Figura 6, se muestra un diagrama de cajas de dos distribuciones de datos: la distribución ideal y la distribución de la métrica WMC en JFreeChart 1.0.1. Se consideran valores altos y bajos a aquellos valores por encima y por debajo del bigote más alto y del bigote más bajo respectivamente. En un proceso ideal los valores fuera de los bigotes (*outlayer*) se deberían eliminar aplicando detección y corrección de *bad smells*.

Como se ha concluido anteriormente, el uso de los mismos umbrales para diferentes productos no parece adecuado y el ajuste de los mismos debería ser un proceso asistido. A continuación se muestra una revisión de uno de los resultados obtenidos en trabajos previos, aplicando todo lo obtenido hasta ahora.

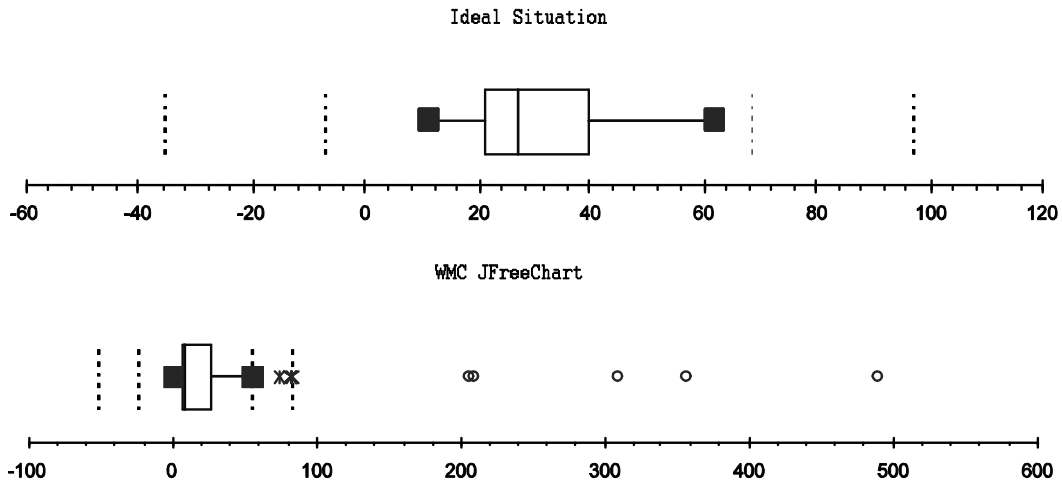


Figura 6 Diagrama de Box Plot

4.1 Detección de un *bad smell*: *Lazy Class*

Revisando de nuevo la definición [4], se trata de "clases que no están haciendo mucho por sí mismas y que podrían ser eliminadas". Los criterios que se establecieron eran un bajo número de métodos y atributos, con complejidad baja y bajo valor de DIT. Además se puede añadir algún criterio adicional como pequeño número de hijos.

El problema de este tipo de planteamientos es: ¿qué se considera bajo en este sistema? Como muy bien se apuntó en [8,10], ciertos sistemas son particulares y puede que las conclusiones generales extraídas no sean correctas. Si se toman como umbrales los tres cuantiles Q1 de NOF, NOM y WMC.
Como se observa existen ciertas discrepancias en los filtros (NOF<=Q1 AND NOM<=Q1 AND WMC<=Q1) a utilizar. Por ejemplo, ¿qué ocurre si aplicamos los valores recogidos en junit a jfreechart como filtro? En este caso para jfreechart se obtienen como sospechosas 63 clases mientras que utilizando los valores propios de jfreechart el número de clases es de 97. La diferencia de número de clases recogida muestra que no se pueden aplicar los mismos criterios sobre uno u otro producto.

4.2 Detección de un *bad smell*: *Large Classes*

Aparentemente, este *bad smell* parece uno de los más sencillos de detectar. Sin embargo, como se ha podido extraer anteriormente, cada sistema puede tener sus particularidades. En concreto, si se establece el mismo umbral para todos, cabe la posibilidad de tomar un valor muy grande que aplicado a muchos productos no devuelva ningún resultado. Puede ocurrir lo contrario y es que al elegir un valor pequeño se tomen demasiados elementos en otros conjuntos.

Utilizando el conocimiento de cómo se distribuyen los datos, se buscan valores entre los que se denominan extremos. En concreto, si se fijan como valor umbral el cuantil Q3 para cada producto y combinando las métricas NOF/NOM/WMC.

Observando los valores, se intuye el problema que supondría aplicar los valores de filtrado de JFreeChart a jcoverage o viceversa. En estos casos nos quedamos con los elementos que están por encima del 75% de elementos en la población. Más concretamente, con aquellos cuyos valores de métricas coinciden en el extremo derecho de sus distribuciones (NOF>=Q3 AND NOM>=Q3 AND WMC>=Q3). Idealmente se debería poder ajustar este valor para localizar aquellos elementos que se denominan *outlayers* de la distribución. Si repetimos el proceso de aplicar, por ejemplo el filtro de junit a jfreechart nos encontramos con 148 clases sospechosas. Por el contrario, aplicando el filtro de jfreechart sólo tenemos 108.

Los resultados demuestran la gran diferencia de aplicar unos u otros umbrales. Aunque obviamente la corrección final venga sujeta a una determinación manual de cuantas clases no son falsos positivos, sí que es evidente la diferencia de escalas.

5. Conclusiones y trabajo futuro

El presente trabajo, ha establecido a partir de un caso de estudio, la idoneidad del uso de umbrales relativos, aún considerando que se pueden seguir utilizando y combinando con valores absolutos. El uso de esta última solución no deja de ser recomendable, de hecho se sigue habilitando dicha solución con los perfiles de métricas, aunque como se ha visto cada producto establece normalmente sus propios límites.

La escasa diferencia de resultados entre distintos tipos de producto indica que no se puede usar esto como discriminante. Es el tamaño del producto el que limita en muchos casos los valores de las métricas. Por otro lado, otras métricas son menos sensibles a estos efectos.

En este trabajo no se ha pretendido obtener con precisión umbrales, ni añadir nuevos métodos de obtención de los mismos, sino comprobar empíricamente la necesidad de uso y el soporte a la definición particular de los mismos para cada tipo de producto. El proceso de detección se debería repetir hasta alcanzar una distribución estable intentando reducir el número de

outlayers. Estos resultados podrían ayudar a asistir el mantenimiento sistemático del software, hasta obtener una versión estable.
Finalmente, se ha extendido la solución ya propuesta para incorporar la declaración de *bad smells*, junto con el *framework* de métricas. Esto permite que la herramienta final sea capaz de asistir al usuario a la hora de establecer los criterios de detección de defectos de una forma más objetiva.

Obviamente, existe un gran número de líneas abiertas:

- Se deben validar los resultados, aumentando el número de productos bajo estudio.
- Se debe comprobar si los resultados están influenciados por el lenguaje de programación.
- Aunque no se ha mencionado, la formación y cultura de los programadores puede influir en que el producto establezca sus límites.

Aclaraciones y agradecimientos

Una versión inglesa de este trabajo fue publicada y presentada previamente por los mismos autores en 10th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, disponible en http://www.inf.usi.ch/faculty/lanza/QAOOSE2006/QAOOSE2006Proc.pdf

La traducción del trabajo ha sido financiada por el Ministerio Español de Ciencia e Innovación a través del proyecto de investigación ROADMAP (TIN2008-05675).

Referencias

1. Francisca Muñoz Bravo. A Logic Meta-Programming Framework for Supporting the Refactoring Process. PhD thesis, Vrije Universiteit Brussel, Belgium, 2003.
2. Shyam R. Chimdaber and Chris F. Kemerer. A metrics suite for object oriented design. IEEE Transactions On Software Engineering, 20:476–493, 1994.
3. Yania Crespo, Carlos L´opez, Raul Marticorena, and Esperanza Manso. Language independent metrics support towards refactoring inference. In 9th ECOOP Workshop on QAOOSE 05 (Quantitative Approaches in Object-Oriented Software Engineering). Glasgow, UK. ISBN: 2-89522-065-4, pages 18–29, jul 2005.
4. Martin Fowler. Refactoring. Improving the Design of Existing Code. Number 0-201-48567-2. Addison-Wesley, 2000.
5. V.A. French. Establishing software metric thresholds. 9th International Workshop on Software Measurement, 1999.
6. Mark Lorenz and Jeff Kidd. Object-oriented software metrics: a practical guide. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.
7. Mika Mäntylä. Bad Smells in Software - a Taxonomy and an Empirical Study. PhD thesis, Helsinki University of Technology, 2003.
8. Mika Mäntylä, Jari Vanhanen, and Casper Lassenius. Bad smells - humans as code critics. In 20th IEEE International Conference on Software Maintenance, 2004.
9. Radu Marinescu. Detecting design flaws via metrics in object-oriented systems. In Proceedings of the TOOLS, USA 39, Santa Barbara, USA, 2001.
10. Radu Marinescu. Measurement and Quality in Object-Oriented Design. PhD thesis, Faculty of Automatics and Computer Science, october, 2002.
11. Raul Marticorena. Analysis and definition of a language independent refactoring catalog. In 17th Conference on Advanced Information Systems Engineering (CAiSE 05). Doctoral Consortium, Porto, Portugal., page 8, jun 2005.http://gnomo.fe.up.pt/ caise/.
12. Tom Tourwè and Tom Mens. Identifying Refactoring Opportunities Using Logic Meta Programming. In Proc. 7th European Conf. on Software Maintenance and Reengineering, pages 91 – 100, Benvento, Italy, 2003. IEEE Computer Society.

Definición Formal de Medidas para Modelos BPMN utilizando OCL

Luis Reynoso[1], Elvira Rolón[2], Marcela Genero[3], Félix García[3], Francisco Ruiz[3], Mario Piattini[3]
[1] Universidad Nacional del Comahue
Buenos Aires 1400, Neuquén, Argentina
e-mail: lreynoso@uncoma.edu.ar
[2]Universidad Autónoma de Tamaulipas
Centro Universitario Tampico-Madero, 89336 Tampico, Tamaulipas, México
e-mail: erolon@uat.edu.mx
[3]Departamento de Tecnologías y Sistemas de Información
Instituto de Desarrollo e Investigación Indra-UCLM – Universidad de Castilla-La Mancha
Paseo de la Universidad Nº 4, 13071 Ciudad Real, España
e-mail: {Marcela.Genero, Felix.Garcia, Francisco.RuizG, Mario.Piattini}@uclm.es

Resumen. *Los modelos de procesos de negocio (BPMN) están cobrando cada vez más relevancia y por ello se esta poniendo cada vez más énfasis en su calidad. Esto nos motivó a definir un conjunto de medidas para la comprensibilidad de los modelos BPMN. Nos centramos en la comprensibilidad, ya que un modelo debe ser entendido correctamente antes de intentar hacer cualquier cambio en él. Estas medidas fueron originalmente definidas de manera informal, utilizando lenguaje natural. Sin embargo, como es bien sabido el lenguaje natural es ambiguo y puede dar lugar a diferentes interpretaciones o a interpretaciones incorrectas de los conceptos capturados por una medida y de la forma de obtener su valor. Para evitar estos problemas, en este artículo presentamos la definición formal de las medidas utilizando OCL (Object Constraint Language) y el metamodelo de BPMN (Business Process Modeling Notation). También se presentarán las principales ventajas y lecciones aprendidas que se obtuvieron una vez concluido el presente trabajo y de la aplicación de OCL para la definición formal de medidas para modelos UML y para expresiones OCL.*

Palabras Claves. *Procesos de Negocio, BPMN, OCL, Medición, Definición Formal*

1 Introducción

En la última década muchas organizaciones se han visto involucradas en entornos comerciales de competitividad y de cambio constante, tanto interna como externamente. Por ello, a menudo tienen que actualizar o modificar sus procesos. Este movimiento de las organizaciones hacia la mejora continua se conoce como el BPR (Business Process Reenginering), iniciativa propuesta por (Hammer y Champy, 1994) en los años noventa. En la actualidad, y gracias al recurso conocido como BPM (Business Process Management), que ha crecido en popularidad en los últimos años, todas las fases del ciclo de vida del proceso se están incluyendo, por lo que reúne la teoría de la gestión y las nuevas tecnologías (Smith y Fingar, 2003).

La relevancia de los procesos de negocio es, pues, la consecución de mayor importancia. Este hecho se evidencia por la aparición de diversos lenguajes para modelar los procesos de negocio. Estos lenguajes son muy diferentes unos de otros, ya que dependiendo de la finalidad para el cual cada uno fue creado, estos estudian los procesos de una manera diferente (Dufresne y Martin, 2003). Entre los lenguajes existentes, se debe prestar especial atención a los siguientes: IDEF 0 (FIPS, 1993), IDEF 3 (Mayer et al., 1995), UML 2.0 (OMG, 2003), y BPMN (OMG, 2006), ya que son los lenguajes más frecuentemente utilizados en la industria.

De los lenguajes antes mencionados, el estándar BPMN proporciona una notación que es muy comprensible para todos los usuarios de negocios (OMG, 2006), lo que ha causado que haya ganado popularidad. El estándar BPMN está definido por la unión de las mejores prácticas dentro de la comunidad de modelado de negocio y estandariza una notación para el modelado del proceso de negocio y la semántica de un diagrama de procesos de negocio (BPD, por sus siglas en inglés Business Process Diagram). La definición de BPDs se debe al hecho de que la gente de negocios esta mucho más familiarizada con la visualización de los procesos de negocio en un formato de diagrama de flujo. BPMN sigue el principio de lectura de la notación de los diagramas de flujo.

Un BPD utiliza los elementos gráficos y la semántica que les da soporte a estos elementos tal como se definen en la especificación (OMG, 2006). La especificación de BPMN define muchos

conceptos semánticos utilizados en la definición de procesos y los asocia con los elementos gráficos, marca, y conexiones. Debido al hecho de que la cantidad de conceptos es amplia, los elementos gráficos se dividen en cuatro categorías básicas de los elementos (ver Fig. 1):

- Objetos de Flujo (los principales elementos gráficos),
- Objetos de Conexión (los medios de conexión para los objetos de flujo),
- Swimlanes o Calles (los medios para agrupar los elementos de modelado) ,
- Artefactos (los cuales proporcionan información adicional acerca del proceso).

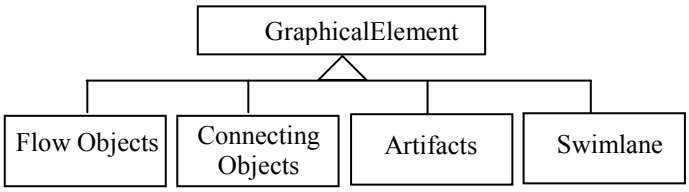


Fig. 1. Principales Elementos Gráficos de BPMN

Estas categorías se dividen a su vez en sub-categorías. Por ejemplo, hay dos formas de agrupar los elementos de modelado principal “swimlanes”, que es mediante el uso de piscinas o Lanes (ver Fig. 2). BPMN especifica todos estos conceptos en detalle, utilizando diagramas de clases para describir la relación entre los elementos gráficos centrales de BPMN, sus atributos, relaciones y tipos.

La creciente importancia de los modelos BPMN ha llevado a diversos autores a centrarse en la calidad de los mismos (Recker et al., 2005; Wohed et al., 2006). En trabajos anteriores hemos definido un conjunto de medidas para la entendibilidad de los modelos expresados en BPMN (Rolón et al., 2006). Con el fin de obtener medidas válidas hemos seguido un método riguroso definido en (Calero et al., 2001) y refinado y extendido en (Reynoso, 2007). El método para la definición de medidas que actualmente se está definiendo y refinando dentro de nuestro grupo de investigación sigue la línea de otras propuestas existentes, tales como la de (Habra et al., 2008). Estas medidas fueron definidas inicialmente en el lenguaje natural (véase la sección 2). Sin embargo una definición en lenguaje natural puede conducirá una interpretación equivocada y esta confusión puede producir efectos indeseables, tales como:

- Las medidas pueden no ser repetibles. Esto significa que dos personas distintas, aplicando la misma medida a un mismo artefacto software, pueden obtener dos resultados diferentes (Kitchenham et al., 1995; ISO, 2001),
- Los resultados experimentales usando la medida pueden ser mal interpretados debido al hecho de que tal vez no sea claro lo que realmente se mide. Esto a su vez, dificulta la réplica experimental (Baroni, 2002).
- Las herramientas utilizadas para la extracción de las medidas pueden obtener diferentes resultados (Baroni, 2002).

Solo una definición formal de las medidas puede evitar muchos de los problemas antes mencionados causados por la definición de medidas imprecisas. Una manera de evitar esto es mediante la definición formal de medidas usando OCL (Object Constraint Languaje) sobre un metamodelo de los artefactos software medidos (Reynoso, 2007). Diversos trabajos se han llevado a cabo en esta área, por ejemplo, en (Reynoso et al., 2006) se presenta la definición formal de medidas para expresiones OCL se define en términos del propio lenguaje OCL, en (Baroni et al., 2002) los autores definen medidas de diagramas de clase y en (Reynoso et al., 2008) se presenta la definición formal de medidas de diagramas de estado.

La definición formal proporciona la definición de medidas en un lenguaje formal, y de acuerdo a la Ontología de la Medición del Software (Garcia et al., 2006), tal definición corresponde al “enfoque de la medición”. La definición formal de una medida se puede realizar una vez que (1)

la medida es definida usando un lenguaje natural, y (2) se ha elegido un metamodelo y un lenguaje formal (estas actividades modeladas en el modelo de actividad de UML, se presentan en la Fig. 3). Además, la definición formal de una medida debería ser coherente con su definición en lenguaje natural. Esto es, la definición en lenguaje natural describe la manera en la cual la medida es obtenida, y por lo tanto su definición formal no debe contradecir esa descripción.

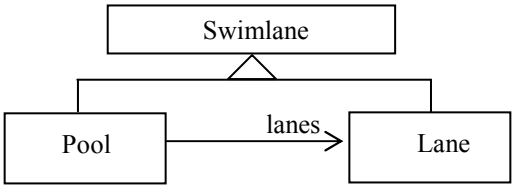


Fig. 2. Swimlane principal

El principal objetivo de este trabajo es definir formalmente las medidas para BPMN utilizando OCL sobre los elementos del metamodelo de BPMN presentado en el Apéndice B, de (OMG, 2007b).

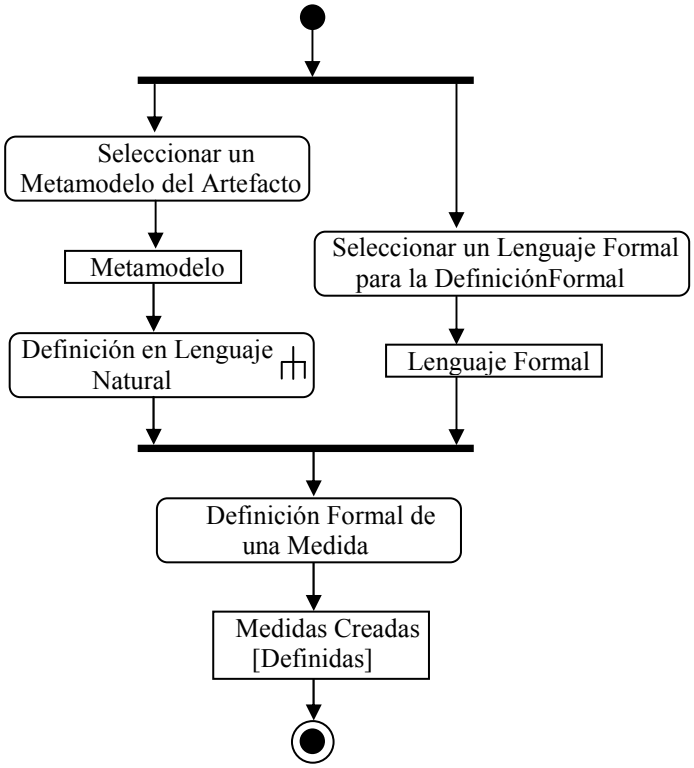


Fig. 3. Principales actividades de la definición de medidas

El resto del artículo está organizado de la siguiente manera: la Sección 2 presenta nuestros trabajos previos, la Sección 3 presenta la definición formal de las medidas para los modelos BPMN. Posteriormente, la Sección 4 se incluye algunas lecciones aprendidas y, por último, la sección 5 esboza las principales conclusiones y el futuro trabajo.

2 Definición Informal

Como ya hemos mencionado, nuestro interés se centra en medir la entendibilidad de los modelos BPMN, pero la entendibilidad es un atributo de calidad externo que sólo se puede medir una vez que loslos modelos se han finalizado. Por esta razón, son necesarias medidas

indirectas para la entendibilidad, centrándose en propiedades estructurales de los modelos BPMN, tales como su complejidad estructural. Posteriormente, es necesaria la validación empírica para evaluar la capacidad de estas medidas de ser utilizadas como indicadores tempranas de la entendibilidad.

Con el objetivo de medir la complejidad estructural de modelos BPMN, hemos propuesto un conjunto de medidas en (Rolón et al., 2006) las cuales están divididas en dos categorías: medidas base y medidas derivadas. La categoría de medidas base consiste de 46 medidas, las cuales cuentan los elementos más significativos del modelo BPMN. Un ejemplo de las medidas base relacionadas a los elementos nodos (gateways), objetos de conexión, calles (swimlanes), artefactos y actividades se muestra en las Tablas 1 y 2. Medidas relacionadas a eventos se describen en los Apéndices. A partir de las medidas base se definieron un conjunto de 15 medidas derivadas con las cuales es posible conocer el conjunto de elementos de una misma categoría, así como las proporciones existentes entre diferentes elementos relevantes del modelo y que pueden pertenecer a varias categorías. El conjunto de medidas derivadas se muestra en la Tabla 3. Una descripción más detallada de las medidas propuestas se presenta en (Rolón et al., 2006).

Elemento Central	Nombre Medida	Definición de la Medida Base	Elemento Central	Nombre Medida	Definición de la Medida Base
Decisión Exclusiva Basada en Datos (XOR)	NEDDB	Número de Decisión/Unión Exclusiva Basada en Datos	Flujo de Secuencia	NSF	Número de Flujos de Secuencia en el proceso
Decisión Exclusiva Basada en Eventos (XOR)	NEDEB	Número de Decisión/Unión Exclusiva Basada en Eventos	Flujo de Mensaje	NMF	Número de Flujos de Mensaje entre Participantes
Inclusiva (OR)	NID	Número de Decisión/Unión Inclusiva	Pool	NP	Número de Pools en el Proceso
Compleja	NCD	Número de Decisión/Unión Compleja	Lanes	NL	Número de Lanes
Paralela (AND)	NPF	Número de Bifurcaciones/uniones Paralelas	Objetos de Datos (Entradas)	NDOIn	Número de Objetos de Datos de entrada a actividades
			Objetos de Datos (Salidas)	NDOOut	Numero de Objetos de Salida

Tabla 1. Medidas Base para Nodos (Gateway), Objetos de Conexión, Calles y Artefactos

Elemento Central	Clasificación	Nombre Medida	Definición de la Medida Base
Tarea	Tarea simple	**NT**	Número de Tareas
	Bucle	**NTL**	Número de Tareas de Bucle
	Instancia Múltiple	**NTMI**	Número de Tareas de Instancia Múltiple
	Compensación	**NTC**	Número de Tareas de Compensación
Sub-Proceso	Sub-Proceso	**NCS**	Número de Sub-Procesos
	Bucle	**NCSL**	Número de Sub-Procesos de Bucle
	Instancia Múltiple	**NCSMI**	Número de Sub-Procesos de Instancia Múltiple
	Compensación	**NCSC**	Número de Sub-Procesos de Compensación
	Ad-Hoc	**NCSA**	Número de Sub-Procesos Ad-Hoc

Tabla 2. Medidas Base para el Elemento Actividad

Nombre de Medida	Definición y Fórmula de la Medida
TNSE	Número Total de Eventos de Inicio del Modelo *TNSE = NSNE+NSTE+NSMsE+NSRE+NSLE+NSMuE*
TNIE	Número Total de Eventos Intermedios del Modelo *TNIE = NINE+NITE+NIMsE+NIEE+NICaE+NICoE+NIRE+NILE+NIMuE*
TNEE	Número Total de Eventos Finales del Modelo *TNEE = NENE+NEMsE+NEEE+NECaE+NECoE+NELE+NEMuE+NETE*
TNE	Número Total de Eventos del Modelo *TNE = NTSE + NTIE + TNEE*
TNT	Número Total de Tareas del Modelo *TNT = NT+NTL+NTMI+NTC*
TNCS	Número Total de Sub-Procesos del Modelo *TNCS = NCS+NCSL+NCSMI+NCSC+NCSA*
TNA	Número Total de Actividades *TNA = TNT + TNCS*
TNG	Número Total de Decisiones/Uniones del Modelo *TNG = NEDDB+NEDEB+NID+NCD+NPF*
TNDO	Número Total de Objetos de Datos en el Modelo *TNDO = NDOIn + NDOOut*
CLA	Nivel de Conectividad entre Actividades $CLA = \frac{TNA}{NSFA}$
CLP	Nivel de Conectividad entre Participantes $CLP = \frac{NMF}{NP}$
PDOPIn	Proporción de Objetos de Datos como productos de entrada y el total de Objetos de Datos $PDOPIn = \frac{NDOIn}{TNDO}$
PDOPOut	Proporción de Objetos de Datos como productos de salida y el total de Objetos de Datos $PDOPOut = \frac{NDOOut}{TNDO}$
PDOTOut	Proporción de Objetos de Datos Producto resultante de las Actividades $PDOTOut = \frac{NDOOut}{TNT}$
PLT	Proporción de Participantes y/o calles y las actividades del Modelo $PLT = \frac{NL}{TNT}$

Tabla 3. Medidas Derivadas para Modelos BPMN

Para ilustrar el cálculo de las medidas definidas para modelos de proceso de negocio se proporciona un ejemplo (ver Fig. 4) que corresponde a un modelo de proceso concurrente de ingeniería para diseñar un chip expresado gráficamente con la notación BPMN. Nuestro objetivo es aplicar las medidas definidas en este trabajo para conocer sus características estructurales. Los valores obtenidos en el cálculo de las medidas base y derivadas se presentan en las Tablas 4 y 5 respectivamente.

Con el objetivo de evaluar cuáles de las medidas definidas pueden ser usadas como indicadores tempranos de la entendibilidad y la modificabilidad se llevaron a cabo dos familias de experimentos. El diseño experimental y los resultados de los experimentos realizados en la primer familia fue descripto en (Rolón et al., 2008). Estos resultados fueron considerados preliminares, ya que no son lo suficientemente concluyentes dado el gran número de medidas propuestas evaluadas inicialmente (60 en total). Sin embargo, estos resultados fueron útiles en la segunda familia de experimentos, donde se seleccionaron las medidas que se consideraron más significativas con respecto a la complejidad estructural de BPM de la primer familia (29 en total). Es posible consultar el diseño y material de la segunda familia de experimentos en http://alarcos.inf-cr.uclm.es/bpmnexperiments. Finalmente, se construyeron modelos de regresión para predecir los tiempos de entendibilidad y modificabilidad, los aciertos y la eficiencia (aciertos/tiempo) de acuerdo a los valores de las medidas (Rolón et al., 2009).

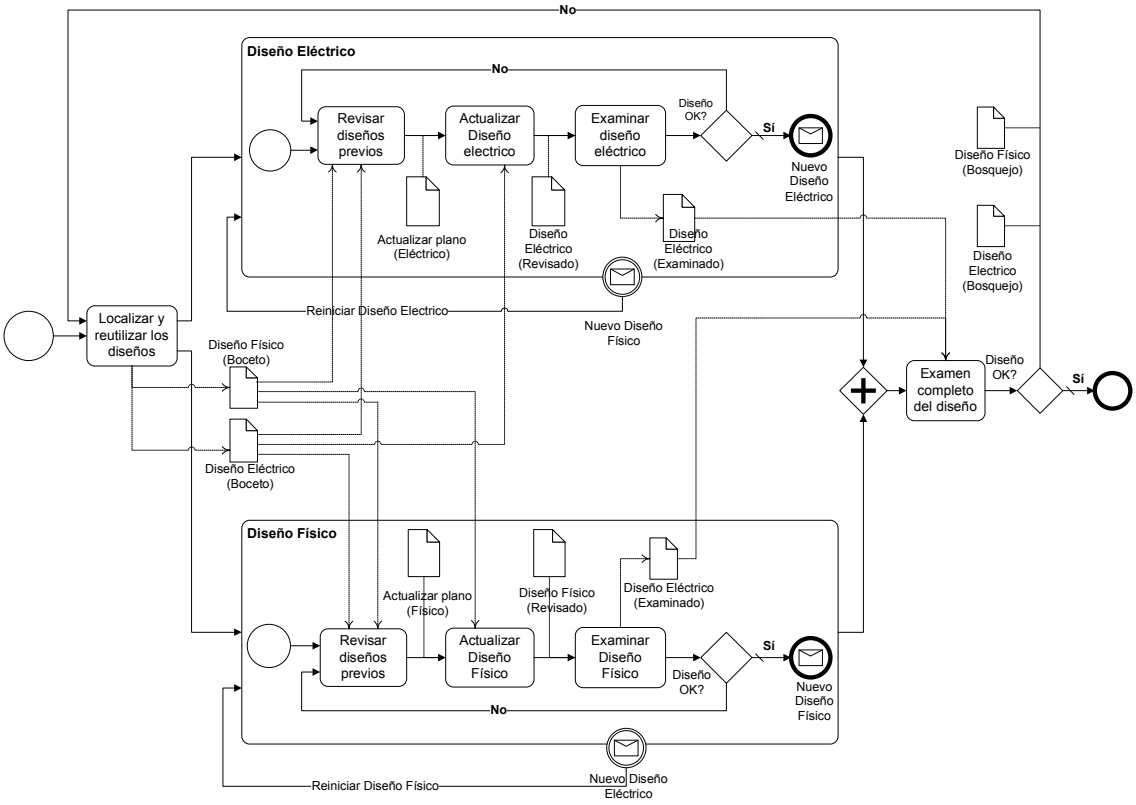


Fig. 4. Ejemplo de Modelo de Proceso de Negocio con BPMN.

Base Measure	Value
NSNE	3
NITE	2
NENE	1
NEMsE	2
NT	8
NEDDB	3
NPF	1
NSF	23
NDOIn	14
NDOOut	8

Tabla 4. Valores de Medidas Base

Derived Measure	Value
TNSE	3
TNIE	2
TNEE	3
TNT	8
TNCS	0
TNE	8
TNG	4
TNDO	22
CLA	8/11 = 0.727
CLP	0
PDOPIn	14/22 = 0.636
PDOPOut	8/22 = 0.363
PDOTOut	8/8 = 1
PLT	2/8 = 0.25

Tabla 5. Valores de Medidas Derivadas

3 Definición Formal

Para la definición formal de las medidas se han definido atributos derivados sobre el diagrama de procesos de negocio (BPD, Business Process Diagram). Por ejemplo, en la Fig. 5 se muestran dos de las medidas definidas (NL y NP) definidas a través de dos atributos derivados los cuales tienen los mismos nombres que las medidas. Una operación de consulta (*getGraphicalElements*) está definida en la metaclase del BPD el cual obtiene los elementos gráficos contenidos en un BPD. Esta operación es definida utilizando una restricción OCL del tipo "definition" en la sección 3.1.

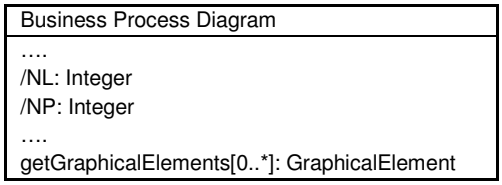


Fig. 5. Agregando atributos y operaciones para la definición de medidas BPMN

3. 1 Operación general para la obtención de los elementos gráficos de BPMN

La Fig. 6 muestra una vista parcial de las principales relaciones entre clases BPMN (OMG, 2007a) las cuales son utilizadas para entender como es definida la operación *getGraphicalElement* de la metaclase del diagrama de procesos de negocio.

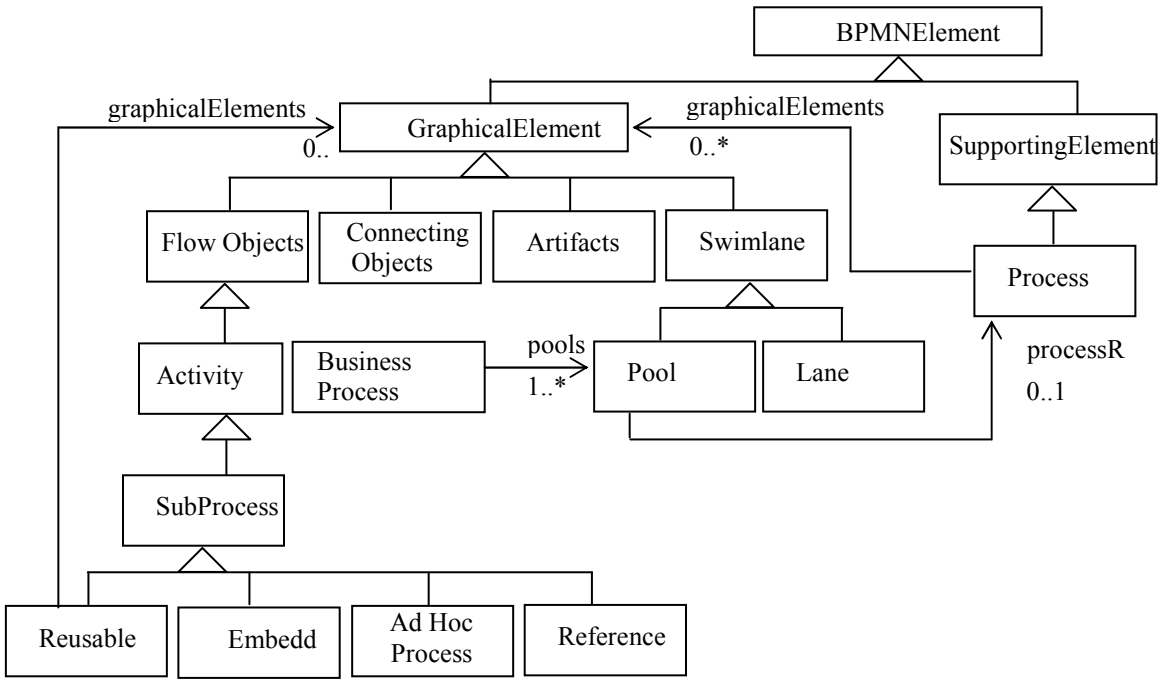


Fig. 6. Vista simplificada de los principales elementos gráficos de BPMN y sus relaciones.

Un diagrama de proceso de negocio expresado con BPMN puede contener uno o más Pools (Calles). A su vez, cada Pool contiene diversos elementos gráficos (eventos, actividades, nodos de decisión/unión, y artefactos). Por lo tanto, para obtener un conjunto de elementos gráficos contenidos en un diagrama de procesos de negocio puede ser definida la siguiente operación:

```
context BusinessProcessDiagram
def: getGraphicalElement() : Set(Graphical Element) = self.pools->
```

```
collect(p:Pool | p)->asSet()->
select (p |p.processRef.notEmpty())->
collect(p|p.processRef.GraphicalElements)->flatten()
```

Sin embargo, la operación previa no toma en cuenta el hecho de que una actividad puede estar *integrada* en un *sub-proceso* el cual puede contener un conjunto de elementos gráficos. Si consideramos ambas situaciones, entonces necesitamos definir el conjunto de elementos gráficos (*getGraphicalElement*) de una manera apropiada:

- En primer lugar, se hace una recopilación del conjunto de elementos gráficos que no son subprocesos integrados, y
- Posteriormente, se añaden los elementos gráficos que forman parte del subproceso.

```
context BusinessProcessDiagram
  def: getGraphicalElement() :Set(Graphical Element) =
  let elements:    Set(GraphicalElement) = self.pools->
collect(p:Pool|p)->asSet()->
select (p |p.processRef.notEmpty())->
collect (p|p.processRef.GraphicalElements)->
flatten() on elements->
  collect (g | not g.oclisType(Embedded) or not g.oclisType(Ad-Hoc Process))->union
  elements->
  collect (g | g.oclisType(Embedded or not g.oclisType(Ad-Hoc Process)))->asSet()->
collect(x |x.getSubProcessElements())->
asSet()) );context GraphicalElement
  def: getSubProcessElements() : Set(GraphicalElement)
  = if self.oclisType(Embedded) then
  self.graphicalelements->
collect(g| g.getSubProcessElements())->asSet()
else
self
  endif
```

3. 2 Definición de las medidas con OCL

La definición formal se presenta de acuerdo a dos aspectos:

- Primero se especifican las medidas base y posteriormente se especifican las medidas derivadas.
- Las medidas base se presentan de acuerdo a la categoría de elementos a la que cada medida está relacionada. De este modo, se presenta la definición formal de las medidas correspondientes a los elementos especificados en (OMG, 2007b) para las categorías de calles (swimlanes), artefactos, objetos de conexión y objetos de flujo.

3.2.1 Medidas Base para Calles (Swimlanes)

En esta sección se describen las medidas relacionadas a los elementos gráficos *calles* o *swimlanes*.

- Medida NL (Número de Lanes). De acuerdo a lo especificado en (OMG, 2007b) dentro de un BPD puede haber uno o más Pools y dentro de un Pool puede haber uno o más Lanes (Figura 7).

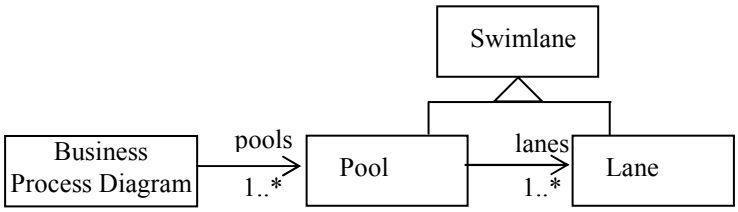


Fig. 7. Relaciones entre Diagrama de Procesos de Negocio, Pools y Lanes

```
context BusinessProcessDiagram
def: NL : Integer = self.pools.lanes-> count()
```

- Medida NP (Número de Pools). En un BPD los participantes están definidos por un Pool, por lo tanto pueden representar un rol o una entidad. Un Pool tiene una asociación con la clase participante, donde un atributo de la clase participante identifica si el participante es un rol o una entidad (Figura 8).

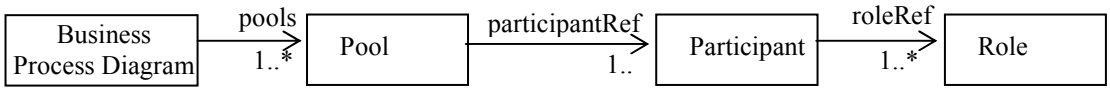


Fig. 8. Relaciones entre Roles y Participantes

```
context BusinessProcessDiagram
def: NP: Integer = self.pools.partipantref->
count(p | p.role->notempty())
```

3.2.2 Medidas Base Elemento Artefactos

- Medida NDOIn (Número de Objetos de datos de entrada). El atributo Conjunto de entrada de la clase Proceso define los requisitos de datos de entrada del proceso. Se pueden definir cero o más Conjuntos de entrada (OMG, 2007b) y cada conjunto de entrada contiene cero o más Artefactos de entrada, los cuales generalmente son objetos de datos (ver Figura 9).

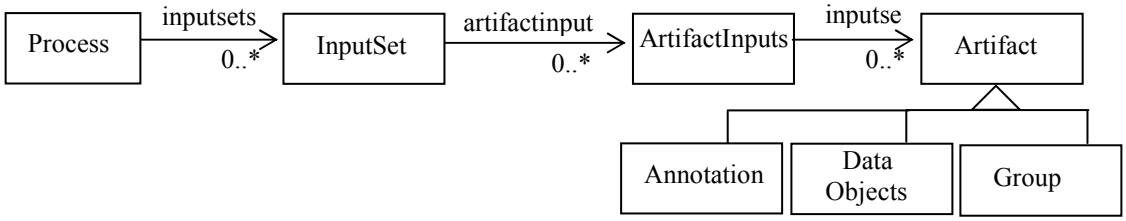


Fig. 9. Objetos de Datos (OMG, 2007b)

```
context BusinessProcessDiagram def: NDOIn : Integer = self.getGraphicalElement()->
collect(e | e.oclisType(Process))->
collect(p.imputsets)->
collect(a|a.artifactinputs.oclisType(Data Object))->
count()
```

- Medida NDOOut (Número de Objetos de datos de salida). De manera similar, el atributo Conjunto de salida de la clase Proceso define los requisitos de las salidas del proceso.

```
context BusinessProcessDiagram def: PDOPOut : Integer = self.getGraphicalElement()->
collect(e |  e.oclisType(Process))->
collect(p.outputsets)->
collect(a | a.artifactref.oclisType(Data Object))->
count()
```

3.2.3 Medidas Base para Objetos de Conexión

En esta categoría se definen formalmente las medidas NSFA, NSFE, NSFG relativas a los flujos de secuencia y la medida NMF (Número de Flujos de mensaje entre participantes) (Tabla 4.12). Las relaciones de estos elementos dentro del conjunto de elementos de un BPD se muestran en la Figura 10.

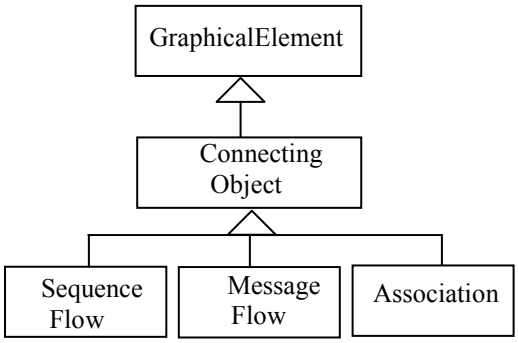


Fig. 10. Principales Objetos de Conexión (OMG, 2007b)

```
context BusinessProcessDiagram
def: NSF : Integer = self.getGraphicalElement()->
count(e |  e.oclisType(Sequence Flow))
```

- Número de Flujos de mensaje entre participantes en el proceso (NMF). Un Flujo de Mensaje es un Objeto de Conexión (ver Fig. 10).

```
context BusinessProcessDiagram def: NMF : Integer = self.getGraphicalElement()->
count(e |  e.oclisType(Message Flow))
```

3.2.4 Medidas Base para Objetos de Flujo

La categoría de Objetos de Flujo se compone por tres subcategorías de elementos que son los Eventos, Actividades y Nodos (Gateways). A continuación se presenta la definición formal de las medidas base, definidas para los elementos que componen cada grupo:

- Los nodos (Gateways) son elementos de modelado que se utilizan para controlar cómo interactúan los Flujos de Secuencia que convergen y divergen dentro de un proceso (OMG, 2007b). Son modelados mediante la jerarquía que se muestra en la Figura 11. Las medidas relacionadas a los Nodos son especificadas formalmente en la Tabla 6.

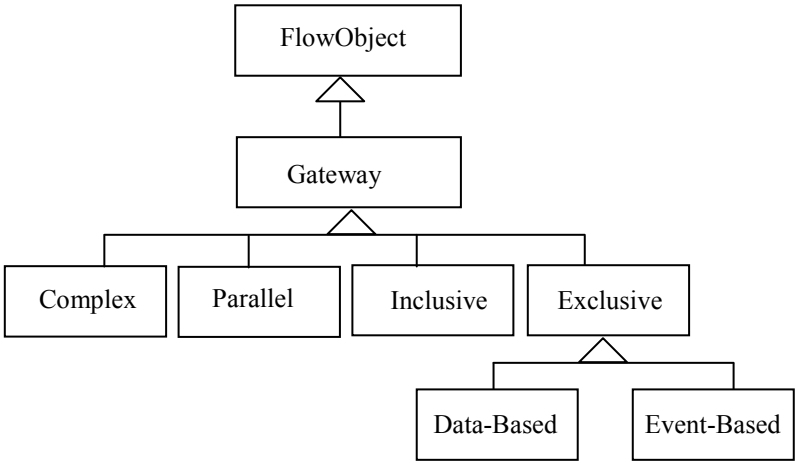


Fig. 11. Objetos de Flujo (OMG, 2007b)

- Medidas para Eventos. Dentro de la categoría Objetos de flujo, se pueden distinguir distintos tipos de eventos (de inicio, intermedios y finales). A su vez, para cada tipo hay diversas modalidades. Las relaciones entre los elementos eventos son modelados en la Figura 12.

Medida	Definición Informal	Definición Formal
NEDDB	Número de Decisión/Unión Exclusiva basada en datos	context BusinessProcessDiagram def: NEDDB : Integer = self.getGraphicalElement()->count(e | e.GatewayType = Exclusive and e.oclisType(Data-based))
NEDEB	Número de Decisión/Unión Exclusiva basada en eventos	context BusinessProcessDiagram def: NEDEB : Integer = self.getGraphicalElement()->count(e | e.GatewayType = Exclusive and e.oclisType(Event-based))
NID	Número de Decisión/Unión Enclusiva	context BusinessProcessDiagram def: NID : Integer = self.getGraphicalElement()->count(e | e.oclisType(Inclusive))
NCD	Número de Decisión/Unión Compleja	context BusinessProcessDiagram def: NCD : Integer = self.getGraphicalElement()->count(e | e.oclisType(Complex))
NPF	Número de Bifurcaciones /Uniones Paralelas	context BusinessProcessDiagram def: NPF : Integer = self.getGraphicalElement()->count(e | e.oclisType(Paralel))

Tabla 6. Medidas base para Nodos (Gateways)

Los valores de los Eventos de Inicio, Intermedio y Final se pueden obtener mediante las siguientes especificaciones:

```
context BusinessProcessDiagram
def: TNSE() : Integer = self.getGraphicalElement->
count(e | e.oclisType(Start) )
def: TNIE() : Integer = self.getGraphicalElement->
count(e | e.oclisType(Intermediate) )
def: TNEE : Integer = self.getGraphicalElement->
count(e | e.oclisType(End) )
```

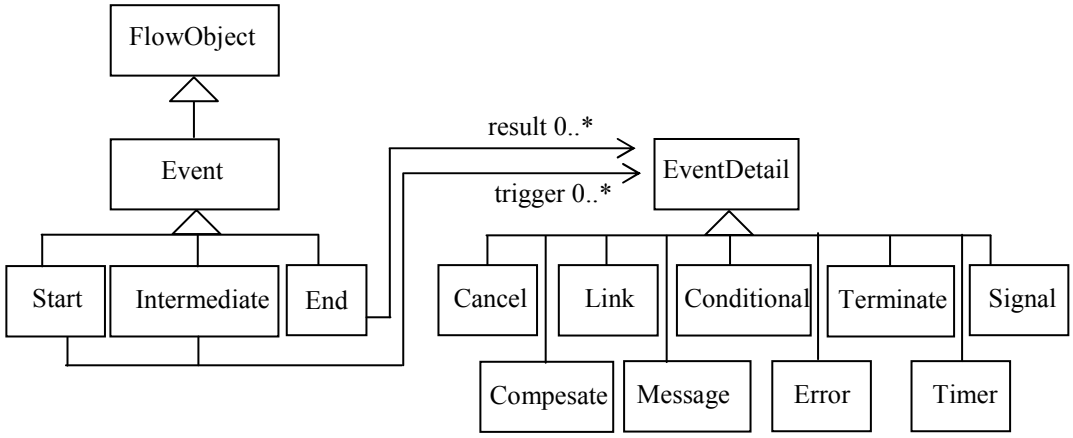


Fig. 12. Relaciones entre Eventos (OMG, 2007b)

De este modo utilizando la especificación anterior la medida derivada TNE es definida de la siguiente manera:

```
context BusinessProcessDiagram
def: TNE : Integer = TNSE + TNIE + TNEE;
```

Sin embargo, es posible obtener el valor de TNSE, TNIE, TNEE en términos de las medidas base definidas en los apéndices A, B y C. Como se puede observar en los Apéndices la especificación utiliza dos importantes atributos: trigger (un atributo que define el tipo de disparador esperado para un evento de Inicio/Intermedio) y result (un atributo que define el tipo de resultado esperado para un Evento Final).

- Medidas para Tareas. Las tareas y subprocesos son los dos elementos que componen esta categoría y son modeladas como una subclase de la clase Actividad (ver Figura 13).

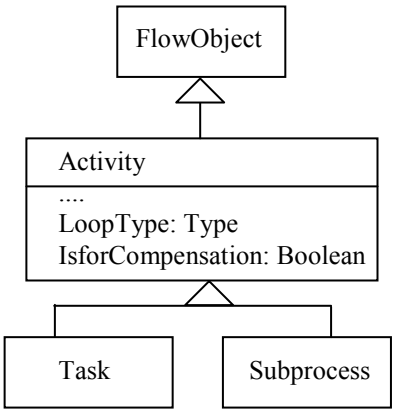


Fig. 13. Actividades (OMG, 2007b)

En la Tabla 7 se presenta la especificación de las medidas para Tareas.

Medida	Definición Informal	Definición Formal
NT	Número de Tareas	`context BusinessProcessDiagram def: NT : Integer = self.getGraphicalElement-> collect(e | e.oclisType(Task) and e.LoopType = None)->count()`
NTL	Número de Tareas de Bucle	`context BusinessProcessDiagram def: NTL : Integer = self.getGraphicalElement-> collect(e | e.oclisType(Task) and e.LoopType = Standard)->count()`
NTMI	Número de Tareas de Instancia Múltiple	`context BusinessProcessDiagram def: NTMI : Integer = elf.getGraphicalElement-> collect(e | e.oclisType(Task) and e.LoopType = Multiple)->count()`
NTC	Número de Tareas de Compensación	`context BusinessProcessDiagram def: NTC : Integer = self.getGraphicalElement-> collect(e | e.oclisType(Task) and e.isforcompensation = true)->count()`

Tabla 7. Medidas para Tareas

- Medidas para Sub-Procesos Colapsados. Subprocesses son una subclase de la clase Activity (ver Fig. 13). Los atributos LoopType y isforCompensation son usados en la especificación de las medidas para subprocesos colapsados (Tabla 8). Los subprocesos son modelados mediante una jerarquía de clases (ver Fig. 14), en la cual las clases reusables son utilizadas para modelar el subproceso colapsado.

Measure	Definición Informal	Formal Definition
NCS	Número de Subprocesos colapsados Simples	context BusinessProcessDiagram def: NCS : Integer = self.getGraphicalElement->collect(e | e.oclisType(Sub-Process))->collect(s | s.subProcessType =reusable and s.looptype = None)->count()
NCSL	Número de Subprocesos colapsados de bucle	**context BusinessProcessDiagram** **def: NCSL : Integer = self.getGraphicalElement->collect(e | e.oclisType(Sub-Process))->** collect(s | s.subProcessType = reusable and s.looptype = standard)->count()
NCSMI	Número de subprocesos colapsados de instancia múltiple	context BusinessProcessDiagram def: NCSMI : Integer = self.getGraphicalElement->collect(e | e.oclisType(Sub-Process))->collect(s | s.subProcessType = reusable and s.looptype = multiInstance)->count()
NCSC	Número de subprocesos colapsados de compensación	context BusinessProcessDiagram def: NCSC : Integer = self.getGraphicalElement->collect(e | e.oclisType(Sub-Process))->collect(s | s.subProcessType = reusable and s.isforcompensation = true)->count()
NCSA	Número de subprocesos colapsados Ad-hoc	context BusinessProcessDiagram def: NCSA : Integer = self.getGraphicalElement->collect(e | e.oclisType(AdHocProcess))->count()

Tabla 8. Medidas para Subprocesos

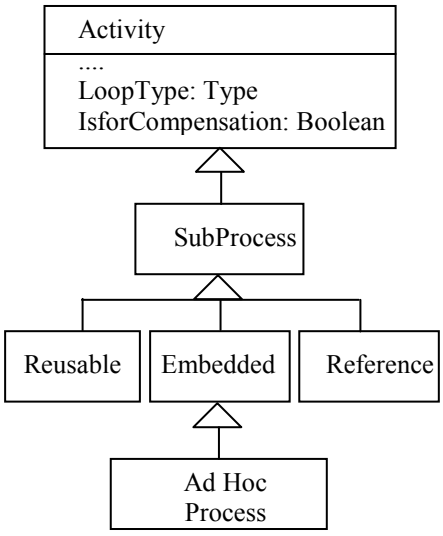


Fig. 14. SubProceso (OMG, 2007b)

3.2.5 Medidas derivadas

La Tabla 9 muestra la especificación de las medidas derivadas.

Medida	Definición Informal	Definición Formal
TNSE	Número total de eventos de inicio del modelo	context BusinessProcessDiagram def: TNSE: Integer = self.NSNE + self.NSTE + self.NSMsE + self.NSRE + … +self.NSMuE
TNIE	Número total de eventos intermedios del modelo	context BusinessProcessDiagram def: TNIE: Integer = self.NINE + self.NITE + self.NIMsE + self.NIEE + self.NICaE + self.NICoE + self.NIRE + self.NILE + self.NIMuE
TNEE	Número total de eventos finales del modelo	Context BusinessProcessDiagram def: TNEE: Integer = self.NENE + self.NEMse + self.NEEE + self.NECaE + self.NECoE + self.NELE + self.NEMuE + self.NETE
TNT	Número total de tareas del modelo	context BusinessProcessDiagram def: TNT : Integer = self.NT + self.NTL + self.NTMI + self.NTC
TNCS	Numero total de subprocesos colapsados del modelo	context BusinessProcessDiagram def: TNCS : Integer = self.NCS + self.NCSL + self.NCSMI + self.NCSC + self.NCSA
TNE	Número total de eventos del modelo	context BusinessProcessDiagram def:TNE : Integer = self.TNSE + self.TNIE + self.TNEE
TNG	Numero total de decisiones/uniones del modelo	context BusinessProcessDiagram def:TNG : Integer = self.NEDDB + self.NEDEB + self.NID + self.NCD +self.NPF
TNDO	Número total de objetos de datos del modelo	context BusinessProcessDiagram def:TNDO : Integer = self.NDOIn + self.NDOOut
CLA	Nivel de conectividad entre actividades	context BusinessProcessDiagram def: CLA: Real = self.TNT.div(self.NSF)
CLP	Nivel de conectividad entre participantes	context BusinessProcessDiagram def: CLP: Real = self.NMF.div(self.NP)
PDOPIN	Proporción de objetos de datos como productos de entrada	context BusinessProcessDiagram def: PDOPIn : Real = self.NDOIn.div(TNDO)
PDOPOUT	Proporción de objetos de datos como productos de salida	context BusinessProcessDiagram def: PDOPOut: Real = self.NDOOut.div(self.TNDO)
PDOTOUT	Proporción de objetos de datos como producto resultante de actividades del modelo	context BusinessProcessDiagram def: PDOTOut: Real = self.NDOOut.div(self.TNT)
PLT	Proporción de participantes y/o calles y las actividades del modelo	context BusinessProcessDiagram def: PLT: Real = self.NL.div(self.TNT)

Tabla 9. Medidas Derivadas

4 Lecciones Aprendidas

Después de llevar a cabo la definición formal de las medidas presentadas, tanto en este artículo como en trabajos previos (Reynoso et al., 2006, 2008) podemos proporcionar las siguientes recomendaciones:

1. El uso de un metamodelo de dominio software durante la actividad de definición de la medida es un aspecto clave a considerar. La definición de una medida tiene que ser lo suficientemente clara y detallada de modo que cualquier concepto del artefactos software (el objeto del estudio) mencionado en su definición en el lenguaje natural pueda ser medible. Para cumplir con este propósito, un metamodelo del artefacto software que se está midiendo debe ser seleccionado, como una actividad previa a la definición de cualquier medida. Como se define en (Jacquet y Abran, 1997), un metamodelo constituye el conjunto de características seleccionadas para representar un software o una pieza de software y el conjunto de sus relaciones, y éstas son propuestas para la descripción del software al cual

se aplicará el método de medición. En consecuencia, la utilización de un metamodelo nos permitirá: (1) examinar si alguno de los conceptos mencionados en la definición de la medida (utilizando el lenguaje natural) es un elemento del metamodelo seleccionado, y (2) definir formalmente cada una de las medidas utilizando un lenguaje formal.

2. Recomendamos OCL como un lenguaje adecuado para la definición formal. OCL se está convirtiendo en el lenguaje *de facto* con el cual se modelan restricciones, ha sido ampliamente utilizado en el modelado de restricciones para el lenguaje UML a través de sus versiones más recientes (por ejemplo, de UML 1.4 a 2,0) y se utiliza para definir las transformaciones en el modelo MDA (Model Driven Architecture). Por otra parte, la definición formal de las medidas utilizando OCL se puede introducir en herramientas compatibles MDA para extraer los valores de las medidas de los modelos UML.

3. Siempre que sea posible, es mejor definir las operaciones genéricas para la definición formal de las medidas. En las propuestas que incluyen definiciones de medidas, es común encontrar que algunas de las medidas están relacionadas entre sí a través de conceptos compartidos. En estos casos es útil definir operaciones genéricas que factoricen operaciones a fin de facilitar la extracción de la medida por medio de herramientas. Por ejemplo, en la Sección 3 se define una operación general a través de la cual se obtienen todos los elementos gráficos dentro de un diagrama de proceso de negocio. Esta operación, llamada *getgraphicalelements*, se reutilizó en varias definiciones de medidas. Es importante factorizar estas operaciones genéricas mediante el uso de los mismos conceptos de abstracción en los que se basa el metamodelo (en el cual las medidas están definidas). Un enfoque similar para definir la operación genérica fue aplicada en la definición formal de las medidas para expresiones OCL (Reynoso et al., 2006) y en la definición de las medidas de diagrama de estado UML (Reynoso et al., 2008).

4. Con la definición formal de las medidas para modelos BPMN fue posible identificar la ambigüedad de algunas definiciones de medidas. Un ejemplo de esto es la medida de NP. Esta medida base consiste en contar el número de piscinas (pools) en un modelo, pero mediante la definición formal de las medidas pudimos comprobar que el concepto de participantes no había sido utilizado en la definición inicial utilizando lenguaje natural, y cuando se define formalmente la medida, es posible distinguir un participante (pool) como una entidad y un rol dentro de un participante (lane).

5 Conclusiones

La principal contribución de este artículo es la definición formal de las medidas para modelos BPMN propuestas en (Rolón et al., 2006) utilizando OCL y el metamodelo de BPMN. Una definición formal de las medidas es útil para obtener medidas repetibles, es decir, medidas que producen el mismo resultado cada vez que se aplican a un mismo artefacto por dos personas diferentes. Por lo tanto, se ven beneficiados los involucrados (“stakeholders”) dentro de un proceso de negocio, junto con todas las personas que usan nuestras medidas BPMN como indicadores tempranos de la entendibilidad de los diagramas de Procesos de negocio, ya que cuentan con una definición precisa de cómo se obtiene el valor de la medida, sin haber malentendidos que puedan ser introducidos al haber utilizado una definición imprecisa en lenguaje natural. Las personas que deseen construir una herramienta de extracción de medidas usando el metamodelo BPMN también se beneficiarán, ya que pueden tomar ventaja de la transformación de la definición formal de las medidas BPMN mediante expresiones de código OCL utilizando herramientas compatibles con MDA.

Como parte de nuestro trabajo futuro, se tiene planeado alinear el trabajo actual usando la última versión de la especificación de BPMN 2 (OMG, 2009) que está siendo desarrollado por el OMG (Object Management Group).

Agradecimientos

La investigación presentada en este artículo es parte de los proyectos MEDUSAS (IDI-20090557) financiado por el "Centro para el Desarrollo Tecnológico Industrial, Ministerio de Ciencia e Innovación" (CDTI), proyecto PEGASO/MAGO (TIN2009-13718-C02-01) financiado por el “Ministerio de Ciencia e Innovación MICINN y Fondo Europeo de Desarrollo Regional FEDER”, los proyectos financiados por la “Consejería de Ciencia y Tecnología de la

Junta de Comunidades de Castilla-La Mancha": INGENIO (PAC 08-0154-9262) y MECCA (PII2I09-0075-8394) y el proyecto 04/E073 financiado por la Universidad del Comahue (Argentina).

Apéndice A

Medidas base para Eventos de Inicio de BPMN

Medida	Definición Informal	Definición Formal
NSNE	Numero de Eventos de inicio simple	context BusinessProcessDiagram def: NSNE: Integer = self.getGraphicalElement->count(e | e.oclisType(Start) and e.trigger->isEmpty())->count()
NSTE	Numero de Eventos de inicio de tiempo	context BusinessProcessDiagram def: NSTE: Integer = self.getGraphicalElement->count(e | e.oclisType(Start) and e.trigger->isnotEmpty() and e.trigger.oclistype(Timer))->count()
NSMsE	Número de Eventos de inicio de mensaje	context BusinessProcessDiagram def: NSMsE: Integer = self.getGraphicalElement->count(e | e.oclisType(Start) and e.trigger->isnotEmpty() and e.trigger.oclistype(Message))->count()
NSLE	Número de eventos de inicio de vínculo	context BusinessProcessDiagram def: NSLE: Integer = self.getGraphicalElement->count(e | e.oclisType(Start) and e.trigger->isnotEmpty() and e.trigger.oclistype(Link))->count()
NSRE	Número de eventos de inicio de regla	context BusinessProcessDiagram def: NSRE: Integer = self.getGraphicalElement->count(e | e.oclisType(Start) and e.trigger->isnotEmpty() and e.trigger.oclistype(Rule))->count()
NSMuE	Número de Eventos de inicio múltiple	context BusinessProcessDiagram def: NSMuE: Integer = self.getGraphicalElement->count(e | e.oclisType(Start) and e.trigger->isnotEmpty() and e.trigger->size() > 1)->count()

Apéndice B

Medidas base para Eventos Intermedios de BPMN

Medida	Definición Informal	Definición Formal
NINE	Número de Eventos intermedios simples	context BusinessProcessDiagram def: NINE: Integer = self.getGraphicalElement->count(e | e.oclisType(Intermediate) and e.trigger->isEmpty())->count()
NITE	Número de Eventos intermedios de tiempo	context BusinessProcessDiagram def: NITE: Integer = self.getGraphicalElement->count(e | e.oclisType(Intermediate) and e.trigger->isnotEmpty() and e.trigger.oclistype(Timer))->count()
NIMsE	Número de Eventos intermedios de mensaje	context BusinessProcessDiagram def: NIMsE: Integer = self.getGraphicalElement->count(e | e.oclisType(Intermediate) and e.trigger->isnotEmpty() and e.trigger.oclistype(Message))->count()
NIEE	Numero de Eventos intermedios de error	context BusinessProcessDiagram def: NIEE: Integer = self.getGraphicalElement->count(e | e.oclisType(Intermediate) and e.trigger->isnotEmpty() and e.trigger.oclistype(Error))->count()
NICaE	Número de Eventos intermedios de cancelación	context BusinessProcessDiagram def: NICaE: Integer = self.getGraphicalElement->count(e | e.oclisType(Intermediate) and e.trigger->isnotEmpty() and e.trigger.oclistype(Cancel))->count()
NICoE	Número de Eventos intermedios de compensación	context BusinessProcessDiagram def: NICoE: Integer = self.getGraphicalElement->count(e | e.oclisType(Intermediate) and e.trigger->isnotEmpty() and e.trigger.oclistype(Compensate))->count()
NIRE	Número de Eventos intermedios de regla	context BusinessProcessDiagram def: NIRE: Integer = self.getGraphicalElement->count(e | e.oclisType(Intermediate) and e.trigger->isnotEmpty() and e.trigger.oclistype(Conditional))->count()

NILE	Número de Eventos intermedios de vínculo	context BusinessProcessDiagram def: NILE: Integer = self.getGraphicalElement->count(e | e.oclisType(Intermediate) and e.trigger->isnotEmpty() and e.trigger.oclistype(Link))->count()
NIMuE	Número de Eventos intermedios múltiple	context BusinessProcessDiagram def: NIMuE: Integer = self.getGraphicalElement->count(e | e.oclisType(Intermediate) and e.trigger->isnotEmpty() and e.trigger->size() > 1)->count()

Apéndice C

Medidas base para Eventos Finales de BPMN

Medida	Definición Informal	Definición Formal

NENE	Número de Eventos finales simples	`context BusinessProcessDiagram`
		`def: NENE: Integer = self.getGraphicalElement->`
		`count(e | e.oclisType(End) and e.result->isEmpty())->count()`

NEMsE	Numero de eventos finales de mensaje	`context BusinessProcessDiagram`
		`def: NEMsE: Integer =`
		`self.getGraphicalElement->count(e | e.oclisType(End) and e.result->isnotEmpty() and e.result.oclistype(Message) )`
		`->count()`

NEEE	Número de eventos finales de error	`context BusinessProcessDiagram`
		`def: NEEE: Integer = self.getGraphicalElement->`
		`count(e | e.oclisType(End) and e.result->isnotEmpty() and e.result.oclistype(Error) )->count()`

NECaE	Número de eventos finales de cancelación	`context BusinessProcessDiagram`
		`def: NECaE: Integer = self.getGraphicalElement->`
		`count(e | e.oclisType(End) and e.result->isnotEmpty() and e.result.oclistype(Cancel) )->count()`

NECoE	Número de eventos finales de compensación	`context BusinessProcessDiagram`
		`def: NECoE: Integer = self.getGraphicalElement->`
		`count(e | e.oclisType(End) and e.result->isnotEmpty() and e.result.oclistype(Compensate) )->count()`

NELE	Numero de Eventos finales de vínculo	`context BusinessProcessDiagram`
		`def: NELE: Integer = self.getGraphicalElement->`
		`count(e | e.oclisType(End) and e.result->isnotEmpty() and e.result.oclistype(Link) )->count()`

NEMuE	Número de eventos finales de vínculo	`context BusinessProcessDiagram`
		`def: NEMuE: Integer = self.getGraphicalElement->`
		`count(e | e.oclisType(End) and e.result->isnotEmpty() and e.result->size() > 1 )->count()`

NETE	Número de eventos finales de terminación	`context BusinessProcessDiagram`
		`def: NETE: Integer = self.getGraphicalElement->`
		`count(e | e.oclisType(End) and e.result->isnotEmpty()and e.result.oclistype(Terminate) )->count()`

Referencias

1. Baroni, A. L. (2002). Formal definition of object-oriented design metrics. Master of Science in Computer.
2. Baroni, A. L., Braz, S. y Brito e Abreu, F. (2002). Using OCL to formalize object-oriented design metrics definitions. 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QUAOOSE'02), Malaga, Spain.
3. Calero, C., Piattini, M. y Genero, M. (2001). Method for obtaining correct metrics. 3rd. International Conference on Enterprise and Information Systems (ICEIS'2001), Setúbal, Portugal pp. 779-784
4. Dufresne, T. y Martin, J. (2003). Process modeling for E-business, George Mason University.
5. FIPS (1993). Integration definition for function modeling (IDEF0), National Institute of Standards and Technology.
6. Garcia, F., Bertoa, M. F., Calero, C., Vallecillo, A., Ruiz, F., Piattini, M. y Genero, M. (2006). Towards a consistent terminology for software measurement. Information and Software Technology 48(8): pp. 631-644.
7. Habra, N., Abran, A., Lopez, M. y Sellami, A. (2008). A framework for the design and verification of software measurement methods. Journal of Systems and Software, 81(5): pp. 633-648.
8. Hammer, M. y Champy, J. (1994). Reengineering the corporation: A manifesto for business revolution. London: Nicholas Brealey.
9. ISO (2001). ISO/IEC, 9126 Software product evaluation- Quality characteristics and guidelines for their use.
10. Jacquet, J. P. y Abran, A. (1997). From software metrics to software measurement methods. 3rd International Software Engineering Standards Symposium (ISESS´97), Washington, DC, USA, IEEE Computer Society, pp. 128-135
11. Kitchenham, B., Pfleeger, S. y Fenton, N. (1995). Towards a framework for software measurement validation. IEEE Transactions on Software Engineering 21(12): pp. 929-944.

12. Mayer, R. J., Menzel, C. P., Painter, M. K., de White, P. S., Blinn, T. y Perakath, B. (1995). Information integration for concurrent engineering (IICE) IDEF3 Process Description Capture Method Report. College Station, Texas.
13. OMG (2003). Unified Modeling Language (UML) Specification: Infrastructure, version 2.0, Object Management Group.
14. OMG (2006). Business Process Modeling Notation (BPMN) Specification, Object Management Group. http://www.bpmn.org/Documents/OMG%20Final%20Adopted%20BPMN%201-0%20Spec%2006-02-01.pdf
15. OMG (2007a). Business Process Definition MetaModel (BPDM), OMG Adopted Specification, Object Management Group.
16. OMG (2007b). Business Process Modeling Notation (BPMN) Specification v 1.1 (draft), Object Management Group.
17. OMG, 2009. Business Process Model and Notation (BPMN), FTF Beta 1 for Version 2.0. Disponible en http://www.omg.org/spec/BPMN/2.0/
18. Recker, J., Indulska, M., Rosemann, M. y Green, P. (2005). Do process modelling techniques get better? A comparative ontological analysis of BPMN. 16th Australaian Conference on Information Systems, Sydney, Australia.
19. Reynoso, L. (2007). A measurement-based approach for assessing the influence of import-coupling on OCL expressions maintainability. Tesis doctoral, Escuela Superior de Informática. Ciudad Real, España, Universidad de Castilla-La Mancha.
20. Reynoso, L., Cruz-Lemus, J. A., Genero, M. y Piattini, M. (2006). OCL2: using OCL in the formal Definition of OCL expression measures. 1st. Workshop on Quality in Modeling QIM co-located with the ACM/IEEE 9th International Conference on Model Driven Engineering Languages and Systems (MODELs´06), Genova, Italy. pp. 95-107
21. Reynoso, L., Cruz-Lemus, J. A., Genero, M. y Piattini, M. (2008). Formal definition of measures for UML statechart diagrams Using OCL. 23rd ACM Symposium on Applied Computing (SAC-SE´08), Fortaleza, Ceará, Brazil, ACM.
22. Rolón, E., Garcia, F., Ruiz, F., Piattini, M., Visaggio, C. A. y Canfora, G. (2008). Evaluation of BPMN models quality: a family of experiments. 3rd International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE´08), Funchal, Madeira. pp. 56-63
23. Rolón, E., Ruiz, F., Garcia, F. y Piattini, M. (2006). Applying software metrics to evaluate Business Process Models. CLEI-Electronic Journal Vol. 9 (1, Paper 5).
24. Rolón, E., Sanchez, L., Garcia, F., Ruiz, F., Piattini, M., Caivano, D. y Visaggio, C. A. (2009). Prediction models for BPMN usability and maintainability. IEEE Conference on Commerce and Enterprise Computing, Vienna (Austria). pp. 383-390
25. Smith, H. y Fingar, P. (2003). Business process management: The Third Wave. USA, Meghan-Kiffer Press.
26. Wohed, P., van der Aalst, W. M. P., Dumas, M., ter Hofstede, A. H. M. y Russell, N. (2006). On the suitability of BPMN for business process modelling. Business Process Management (BPM´06), Vienna, Austria, Springer-Verlag Berlin Heidelberg. pp. 161-176

Nuevos Libros

Bundschuh, M.; Dekkers, C.:
The IT Measurement Compendium
Springer-Verlag, 2008 (643 páginas)
ISBN 978-3-540-68187-8

El primer paso hacia el éxito en un proyecto software es asegurar un desarrollo profesional. Esto incluye un proceso formal de estimación basado en métricas para asegurar unos fundamentos sólidos en la planificación del proyecto. Estimaciones precisas requieren medidas cuantitativas, idealmente basadas en herramientas. Además, los gestores de proyectos software deben monitorizar y actualizar esas estimaciones durante el ciclo de vida del proyecto para controlar el progreso y evaluar posibles riesgos.

Basándose en sus muchos años de experiencia práctica como gestores de software y consultores, Manfred Bundschuh y Carol Dekkers presentan un marco valioso para cualquiera relacionado con la gestión de proyectos software. Presentan los cinco métodos de medición funcional del software reconocidos por ISO/IEC, con variantes, experiencias, reglas de conteo y casos de estudio, y utilizan numerosos ejemplos prácticos de cómo producir estimaciones realistas.

Escrito con un estilo altamente práctico, incluyendo listas de control, formularios, advertencias de uso y muchos enlaces a organizaciones nacionales e internacionales de medición, este libro es el compañero ideal para los ocupados gestores de proyectos software o para los gestores de aseguramiento de la calidad.

Jones, C.:
Estimating Software Costs: Bringing Realism to Estimating
Mc Graw Hill, 2007 (644 páginas)
ISBN 978-0-07-148300-1

Consiga una clara y completa comprensión de cómo estimar los costes, el calendario y la calidad del software utilizando la información del mundo real contenida en este volumen. Encuentre cómo elegir las herramientas hardware and software correctas, desarrolle estrategias de evaluación, pruebas y prototipos, y produzca estimaciones correctas de los costes del software. Además podrá encontrar una cobertura total de las últimas aproximaciones a las estimaciones en desarrollos que utilicen Java, métodos de orientación a objetos y componentes reutilizables.

Este libro presenta las bases para planificar estimaciones en los niveles de proyecto, fase o actividad; para compensar las inexactitudes que se tengan en la recogida de datos, los cálculos o el análisis de los mismos; calcular los entregables del software y la complejidad de los datos; probar los principios de diseño y las características operacionales utilizando prototipos software; trabajar con los cambios de configuración, investigación, control de calidad, y costes de la documentación.

Próximas Conferencias

IWSM/Mensura/Matrikon/MAIN 2010
10-12 Noviembre 2010
Vector Consulting
Stuttgart, Alemania

Tema y Alcance

Las mediciones y métricas del software son tecnologías clave para manejar y controlar el desarrollo de los proyectos software. La medición es esencial para cualquier actividad de ingeniería y para incrementar el conocimiento científico y técnico teniendo en cuenta tanto el desarrollo práctico del software como la investigación empírica en la tecnología del software. Esta conferencia facilitará el intercambio de experiencias entre la teoría y la práctica de la medición del software.

Temas de Interés

Se anima a enviar contribuciones en cualquier campo de la medición del software, incluyendo, pero no limitándose a:

Fundamentos de métricas del software
Aplicaciones prácticas de la medición
Procesos y recursos de medición
Casos de estudio empíricos
Mediciones del tamaño funcional
Estimación del software
Mejora de procesos software
Métricas de procesos y productos
Evaluación de mejores prácticas basándose en mediciones
Gestión de proyectos basándose en mediciones
Bases de datos de medición
Validación de métricas
Servicios y herramientas de medición
Experiencias de medición
Teoría de la medición
Paradigmas de medición
Comparación de software

Envíos

Los autores deberían enviar *short papers* (2 a 4 páginas) no después del **21 de junio de 2010** a

metrikonpapers@dasma.org

Los artículos no deberían haber sido ya publicados en ningún otro sitio, ni haber sido enviados a una revista o a otra conferencia. Al menos uno de los autores de cada artículo aceptado debe registrarse en la conferencia y asegurar que el artículo será presentado. Los artículos finales tienen que estar basados en el formato MetriKon que se puede obtener en IWSM/MetriKon 2010 guidelines en www.metrikon.de

Los lenguajes de la conferencia serán Alemán e Inglés.

Procesos y Métricas en la WWW

En esta sección de la revista se presenta una lista ordenada de sitios web en los que se tratan los temas de interés de los lectores de la misma.

Sitios Web de Asociaciones Nacionales de Medición del Software

Alemania. Asociación Alemana de Medición del Software. DASMA. www.dasma.org
Finlandia. Asociación Finlandesa de Métricas del Software. FISMA. www.sttf.fi
Holanda. Asociación Holandesa de Métricas del Software. NESMA. www.nesma.nl
Reino Unido. Asociación de Métricas del Software del Reino Unido. UKSMA. www.uksma.co.uk

Sitios Web de Organismos Internacionales de Medición del Software

COmmon Software Measurement International Consortium. COSMIC. www.cosmicon.com
International Function Points Users Group. IFPUG. www.ifpug.com
International Software Benchmarking Standards Group. ISBSG. www.isbsg.org.au

Sitios Web de Laboratorios de Investigación en Medición del Software

Alemania. Laboratorio de Medición del Software. SMLAB. ivs.cs.uni-magdeburg.de/sw-eng/us
Canada. Laboratorio de Investigación en Ingeniería del Software. GELOG. www.gelog.etsmtl.ca
España. Laboratorio de Medición del Software. CuBIT. www.cc.uah.es/cubit

Relación con RPM

Guía para Autores de Artículos de Divulgación

Los artículos de divulgación podrán ser publicados por cualquier persona que pertenezca a una organización miembro de AEMES. Con la pertinente autorización de su organización. Deberán versar sobre algún asunto de interés relacionado con el alcance de AEMES. Los artículos no tendrán revisión por pares pero no podrán ser artículos de información meramente comercial.

Los autores deberán enviar los artículos electrónicamente utilizando la dirección de correo electrónico rpm@aemes.org. Por favor dirigir los artículos al Editor de la Revista de Procesos y Métricas de las Tecnologías de la Información. El artículo debe ser enviado para el proceso de revisión en formato Microsoft Word.

Guía para Autores de Artículos de Investigación

Los artículos de investigación podrán ser publicados por cualquier persona que pertenezca a una organización miembro de AEMES. Deberán versar sobre algún asunto de interés relacionado con el alcance de AEMES.

Los autores deberán enviar los artículos electrónicamente utilizando la dirección de correo electrónico rpm@aemes.org. Por favor dirigir los artículos al Editor de la Revista de Procesos y Métricas de las Tecnologías de la Información. El artículo debe ser enviado para el proceso de revisión en formato Microsoft Word.

El envío de un artículo implica que el trabajo descrito no ha sido publicado previamente (excepto en el caso de una tesis académica), que no se encuentra en ningún otro proceso de revisión, que su publicación es aceptada por todos los autores y por las autoridades responsables de la institución donde se ha llevado a cabo el trabajo y que en el caso de que el artículo sea aceptado para su publicación, el artículo no será publicado en ninguna otra publicación en la misma forma, ni en Español ni en ningún otro idioma, sin el consentimiento de AEMES.

Una vez recibido un artículo se enviará al autor de contacto por correo electrónico un acuse de recibo.

Todos los artículos de investigación recibidos para ser considerados para su publicación serán sometidos a un proceso de revisión. La revisión será realizada por dos o, en su caso, tres expertos independientes. Para asegurar un proceso de revisión lo más correcto posible los nombres de los autores y los revisores permanecerán confidenciales. Una vez revisado un artículo se enviarán por correo electrónico los resultados de la revisión. En el caso de que el artículo haya sido rechazado se adjuntarán las valoraciones de los revisores. El proceso de revisión está libre de costes para los autores.

Una vez que un artículo haya sido aceptado, se solicitará a los autores que transfieran los derechos de autor del artículo a AEMES. Recibida la transferencia, se solicitará a los autores el envío de una versión del artículo lista para publicación que se deberá enviar en formato Microsoft Word.

La publicación de un artículo en la revista está libre de costes para los autores, pero todas las instituciones de origen de todos los firmantes del artículo deberán ser miembros de AEMES.

Guía para la preparación de manuscritos

El texto deberá estar escrito en un correcto castellano (Uso Español) o en Inglés (Uso Británico). Excepto el abstract que deberá estar escrito en un correcto Inglés (Uso Británico).

Abstract y Resumen. Se requiere un abstract en inglés con un máximo de 200 palabras. El abstract deberá reflejar de una forma concisa el propósito de la investigación, los principales y resultados y las conclusiones más importantes. No debe contener citaciones. Se debe

presentar a continuación del abstract en inglés una traducción del mismo al castellano bajo el epígrafe Resumen.

Palabras clave. Inmediatamente después del Resumen se proporcionarán un conjunto de 5 palabras clave evitando términos en plural y compuestos, tampoco se deben usar acrónimos o abreviaturas a no ser que sean de un uso ampliamente aceptado en el campo del artículo. Estas palabras claves serán utilizadas a efectos de indexación.

Subdivisión del artículo. Después del Abstract y el Resumen, que no llevarán numeración, se debe dividir el artículo en secciones numeradas, comenzando en 1 y aumentando consecutivamente. Las subsecciones se numerarán 1.1 (1.1.1, 1.1.2, etc.), 1.2, etc. No se deben incluir subdivisiones por debajo del tercer nivel (1.1.1). Cada sección o subsección debe tener un título breve que aparecerá en una línea separada.
Apéndices. Si hay más de un apéndice, se deben identificar como A, B, etc. Las ecuaciones en los apéndices tendrán una numeración separada: (Eq. A.1), (Eq. A.2), etc.
Agradecimientos. Se deben situar antes de las referencias, en una sección separada.
Tablas. Se deben numerar las tablas consecutivamente de acuerdo con su orden de aparición en el texto. Se deben poner títulos a las tablas debajo de las mismas.
Figuras. Se deben numerar las figuras consecutivamente de acuerdo con su orden de aparición en el texto. Se deben poner títulos a las figuras debajo de las mismas.
Referencias. Se debe verificar que cada referencia citada en el texto se encuentra también en la lista de referencias y viceversa. Los trabajos no publicados o en proceso de revisión no pueden ser citados.
-Citaciones en el texto: Un solo autor. El primer apellido del autor, seguido de una coma y la primera inicial, seguida de un punto, a continuación, tras una coma, el año de publicación. Todo entre corchetes. Dos o más autores. Los nombres de los autores, siguiendo el formato de un solo autor, separados por puntos y comas y el año de publicación. Lista. Las listas deberán ser ordenadas, primero de forma alfabética y luego, si fuera necesario, de forma cronológica. Si hay más de una referencia del mismo autor en el mismo año deben ser identificadas por las letras “a”, “b”, etc., situadas después del año de su publicación.
-Referencias. Véase Volumen 1 Número 1 de esta publicación. Apartado 2.8.2.

Formato

Los autores deberán bajar de la página web de RPM en el sitio web de AEMES el artículo de ejemplo y seguir estrictamente el mismo formato.

ASOCIACIÓN ESPAÑOLA DE MÉTRICAS DE SISTEMAS INFORMÁTICOS AEMES

Facultad de Informática de la UPM. Campus de Montegancedo. 28660-Boadilla del Monte (Madrid)
Teléfono: 91 336 66 08. Fax: 91 336 74 12. E-mail: admon@aemes.org

FORMULARIO DE INSCRIPCIÓN A AEMES

Información Personal
Apellidos: Nombre:
Teléfono: e-mail:
Dirección Personal.
Empleo:

Inscripción Institucional o Empresarial
Institución o Empresa: C.I.F.:
Dirección:

Datos Bancarios
Número de Cuenta.
Titular:

Autorizo a AEMES a cargar a la cuenta arriba indicada la cuota Anual de inscripción que asciende a 360 € + la cuota de inscripción que asciende a 150 €

Fecha y Firma

FORMULARIO DE INSCRIPCIÓN A RPM (Revista de Procesos y Métricas)

Información Personal
Apellidos: Nombre:
Teléfono. e-mail.
Dirección Personal:
Empleo:

Inscripción Institucional o Empresarial
Institución o Empresa: C.I.F.:
Dirección:

Datos Bancarios
Número de Cuenta:
Titular:

Autorizo a AEMES a cargar a la cuenta arriba indicada la cuota Anual de subscripción a RPM que asciende a 60 € + Gastos de Envío

Fecha y Firma