Un Marco para la Definición de Métricas sobre Modelos de Dependencias entre Actores

Xavier Franch, Gemma Grau, Carme Quer

Universitat Politècnica de Catalunya (UPC)
C/Jordi Girona 1-3, UPC-Campus Nord (OMEGA), Barcelona (Spain)
{franch, ggrau, cquer}@lsi.upc.edu

Resumen. Los modelos de dependencias entre actores son un formalismo que describe los procesos como una red de relaciones de dependencias entre actores. En la actualidad, este tipo de modelos se usa, entre otros, en las fases preliminares del análisis de requisitos, en el análisis organizacional y en la reingeniería de procesos. En este artículo, proponemos un marco para la definición de métricas sobre este tipo de modelos, con el objetivo de facilitar el análisis de ciertas propiedades de los procesos tales como la seguridad, la eficiencia o la precisión. Las métricas se definen en términos de los actores y las dependencias del modelo, distinguiendo tres tipos de métricas diferentes que se definen formalmente y se aplican sobre el caso de estudio de un sistema organizador de reuniones.

1. Introducción

Los métodos y lenguajes de análisis orientados a objetivos tales como KAOS, *i**, GRL o Tropos [1, 2, 3] son extensamente usados en la comunidad de la ingeniería de requisitos, pues proporcionan una descomposición y refinamiento de las necesidades del usuario en objetivos concretos que resulta muy útil en las fases preliminares del análisis de requisitos.

Los modelos orientados a actores extienden el modelado de objetivos, permitiendo modelar el sistema mediante una red de actores y dependencias y su descomposición en conceptos simples. Un ejemplo de este tipo de modelos es el modelo estratégico de dependencias (SD) de *i** [2], cuyo formalismo permite expresar una primera vista organizacional del sistema que puede ser analizada a diferentes niveles. Por ejemplo, mediante la exploración de alternativas y la aplicación de métodos de razonamiento respecto a propiedades como la oportunidad y la vulnerabilidad [2].

En este artículo proponemos realizar el análisis estructural de los modelos de dependencias entre actores mediante la definición de métricas adecuadas que permitan la evaluación de los modelos respecto a ciertas propiedades. En otros tipos de modelos ya se han aplicado métricas para analizar su suficiencia. En los modelos orientados a objetos [4, 5], por ejemplo, se usan métricas para medir propiedades estructurales como la cohesión y el acoplamiento. En los modelos de la arquitectura

del sistema [6], surgen las propiedades relativas a los requisitos no funcionales u organizacionales del propio sistema, tales como seguridad, eficiencia y coste.

Nuestra propuesta consiste en la definición y evaluación de ciertas propiedades de los sistemas cuando estos son modelados en lenguajes orientados a actores. Dichas propiedades se evalúan mediante métricas definidas en términos de los actores y dependencias del modelo. Este enfoque facilita el análisis de los requisitos no funcionales y organizacionales del sistema, pues permite expresar las propiedades en términos más cercanos al universo de discurso (objetivos, dependencias…) que a la estructura del sistema (componentes, nodos, conectores…). Para definir el marco general formal para este tipo de métricas, caracterizamos el concepto de modelo de dependencias entre actores y proponemos tres diferentes tipos de métricas. Dependiendo de si el elemento de interés es un actor o una dependencia, obtenemos formas generales de estos tres tipos de métricas basadas en actores o en dependencias.

En la sección 2 introducimos la noción de modelos de dependencias entre actores y damos sus definiciones formales. En la sección 3 identificamos tres categorías de métricas y damos la forma general de cada una de ellas. En las secciones 4 y 5, estudiamos la aplicación del marco propuesto en el caso de los sistemas de planificación de reuniones [7, 8]. Finalmente, en la sección 6, presentamos las conclusiones y el trabajo futuro.

2. Una definición de modelos de dependencias entre actores

Un modelo de dependencias entre actores contiene dos tipos de elementos: los actores y las dependencias entre ellos. Los actores son entidades intencionales de manera que existe un motivo racional detrás de cada actividad que realizan. Las dependencias conectan los actores fuente y objetivo, llamados *depender* y *dependee* respectivamente. De esta manera se forma una red de conocimiento que permite entender "por qué" el sistema se comporta de una forma determinada [9]

Consideramos dos tipos de actores: roles (*roles*) y agentes (*agents*). Según [2], un rol es una caracterización del comportamiento de un actor social en un determinado contexto o dominio, mientras que un agente es un actor con manifestaciones concretas y físicas que juega un determinado rol. Los actores y las dependencias pueden tener atributos, que pueden ser generales (fabricante del producto, localización física...) u orientados a su uso durante las actividades de ingeniería de requisitos (prioridad, importancia…).

Para nuestro objetivo, es útil considerar que los agentes y las dependencias pertenecen siempre a un determinado tipo de elementos con sus mismas características. Respecto a los roles, permitimos que pertenezcan a ningún, uno o más tipos, pues en algunos casos se requiere más flexibilidad. Por ejemplo, el rol de planificador de reuniones puede ser cubierto por agentes humanos o software por lo que su tipo puede ser humano o software.

La definición 1 formaliza el concepto de modelo de dependencias entre actores. En el resto del artículo no consideraremos el caso de dependencias con múltiples *dependers* o *dependees,* pues su tratamiento es fácilmente extrapolable pero dificultaría la comprensión de los conceptos.

Definición 1. Modelo de dependencias entre actores.

Un modelo de dependencias entre actores es un par M = (A, D), siendo A un conjunto de actores y D las dependencias entre ellos, de manera que:

1) El conjunto A permite el mapeo $type_A$: $A \rightarrow$ {rol, agente} que clasifica el modelo de actores entre roles y agentes. Cuando sea conveniente, consideraremos A como un par $A = (A_{rol}, A_{agent})$.
2) El conjunto A permite el mapeo $sort_A$: $A \rightarrow P(TA)$, donde TA es el conjunto de actores permitido. De esta forma: $type_A(a) = agent \Rightarrow \| sort_A(a) \| = 1$
3) Los actores en el conjunto A pueden tener atributos modelados como mapeos $\{ prop_{A;i}: A \times K_i \rightarrow V_i \}$.
4) D se define como un conjunto de pares de actores ordenados con el nombre de la dependencia $D \subseteq A \times A \times string$.
5) El conjunto D permite el mapeo $sort_D$: $D \rightarrow TD$, siendo TD el conjunto de tipos de dependencias permitido.
6) Las dependencias en el conjunto D pueden tener atributos modelados como mapeos $\{ prop_{D;i}: D \times K_i \rightarrow V_i \}$.

Para presentar los ejemplos, usaremos los modelos de Dependencias Estratégicas (SD) de *i** [2]. Los modelos SD permiten 4 tipos de dependencias distintas: objetivo (*goal*), tarea (*task*), recurso *(resource*) y requisitos no funcionales (*softgoals*).

3. Métricas para los modelos de dependencias de actor

Las métricas estructurales se usan para el análisis de aquellas propiedades de un modelo de dependencias de actor que dependen de la forma del modelo y del tipo, orden y atributos de sus elementos. Este tipo de métricas son útiles tanto para el análisis del modelo abstracto del sistema (compuesto básicamente por roles), como para la comparación de las diferentes instancias de este modelo abstracto (compuestos básicamente por agentes) con respecto a aquellos criterios más relevantes.

Dada una propiedad del modelo que se quiera medir, puede pasar que todos sus elementos (actores y dependencias) influencien la métrica. Sin embargo, es mucho más probable que sólo unos pocos elementos de un tipo particular afecten esta propiedad. De aquí en adelante, algunos elementos individuales serán identificados como relevantes para la propiedad y además, de manera general, todos los elementos tendrán pesos diferentes en la métrica. Estas consideraciones se adoptan para permitir un marco de aplicación más amplio de las métricas formuladas.

Distinguimos tres tipos diferentes de métricas estructurales. Las métricas estructurales globales toman el modelo como un todo y producen una única medición para la propiedad de interés. En cambio, las métricas estructurales locales se centran en los elementos individuales del modelo, produciendo un conjunto de valores que puede ser examinado para encontrar los puntos débiles y/o fuertes del modelo. Llamaremos métricas sensitivas a las métricas estructurales locales que permiten encontrar aquellos elementos que maximizan los valores de una métrica estructural local. Todos los tipos de métricas están basadas en dos conceptos fundamentales: la evaluación de actores y la evaluación de dependencias.

3.1. Evaluación de actores y dependencias

El marco de métricas propuesto tiene como concepto atómico la evaluación de los elementos individuales que hay en el modelo de dependencias entre actores. Asimismo, la evaluación de los actores y las dependencias están definidas como funciones que dan valores en el intervalo [0, 1]. Dicha evaluación se calcula siempre como la multiplicación de dos factores, pero en la evaluación de actores se tienen en cuenta sólo el tipo, clase y atributos del propio elemento (dependencias que salen o entran del actor), mientras que en la evaluación de las dependencias se consideran, además del propio elemento, aquellos elementos que están relacionados con él (actores asociados a la dependencia).

Definición 2. Evaluación de actores.

Sea P una propiedad, M = (A, D) un modelo de dependencias entre actores y $a \in A$ un actor del modelo, la evaluación del actor a para P sobre M, será de la forma:

$$P_{M,A}(a) = f_{M,P}(a) \times g_{M,P}(a))$$

donde $f_{M,P}$: $A \rightarrow [0, 1]$ es un mapeo que asigna un peso a cada actor del modelo, y $g_{M,P}$: $A \times D \rightarrow [0, 1]$ es un mapeo que corrige este peso en un actor considerando las dependencias que entran o salen de él.

Definición 3. Evaluación de dependencias.

Sea P una propiedad, $M = (A, D)$ un modelo de dependencias entre actores y $d=(a, b, x) \in D$ una dependencia del modelo, la evaluación de la dependencia *d* para *P* sobre *M* es de la forma:

$$P_{M,D}(d) = f_{M,P}(d) \times g_{M,P}(d)$$

donde $f_{M,P}$: $D \rightarrow [0, 1]$ es un mapeo que asigna un peso a cada dependencia del modelo y $g_{M,P}$: $D \rightarrow [0, 1]$ es un mapeo que corrige el peso de la dependencia considerando los actores *depender* y *dependee*, respectivamente.

3.2. Métricas estructurales globales

Las definiciones 2 y 3 proporcionan un marco genérico para la definición de las métricas estructurales globales que, dependiendo del tipo de elemento que se requiera enfatizar, pueden ser basadas en actores o basadas en dependencias. Las métricas suman las evaluaciones de sus elementos y hacen una normalización final del valor, que notamos $limit_P$, teniendo en cuenta el número de actores o dependencias del sistema que satisfacen una condición particular. Algunas veces el valor puede considerarse no normalizable y, entonces, la función $limit_P$ tiende a 1.

Definición 4. Métricas estructurales globales basadas en actores.

Sea P una propiedad, $M = (A, D)$ un modelo de dependencias entre actores y $limit_P$: $A \rightarrow [1, ||A||]$ una función, una métrica global estructural basada en actores para P sobre el modelo M es de la forma:

$$P_M = \frac{\Sigma a \in A: \mathrm{P}_{M,A}(a)}{\mathrm{limit}_P(A)}$$

Definición 5. Métricas estructurales globales basadas en dependencias.

Sea P una propiedad, $M = (A, D)$ un modelo de dependencias entre actores y $limit_P$: $D \rightarrow [1, ||D||]$, una función, una métrica global estructural basada en dependencias para P sobre M es de la forma:

$$P_M = \frac{\Sigma d \in D: \mathrm{P}_{M,D}(d)}{\mathrm{limit}_P(D)}$$

3.3. Métricas estructurales locales y sensitivas

La definición de las métricas estructurales locales es parecida a la de las métricas globales pero focaliza en la medición de los elementos individuales. Usamos este tipo de métricas para obtener un análisis completo de los actores y dependencias individuales del modelo. Las métricas sensitivas enfatizan este concepto, proporcionando el valor máximo de las métricas locales estructurales.

Definición 6. Métricas locales estructurales y sensitivas basadas en actores.

Sea P una propiedad, $M = (A, D)$ un modelo de dependencias entre actores y $limit_P$: $A \rightarrow [1, ||A||]$ una función, una métrica local estructural basada en actores para P sobre el modelo M es de la forma:

$$P_M: A \rightarrow [0, 1] \text{ tal que } \mathrm{P}_M(a) = \mathrm{P}_{M,A}(a)$$

Definimos las métricas locales sensitivas basadas en actores fijando P como:

$$PMax_M = \max a \in A: \mathrm{P}_M(a)$$

Definición 7. Métricas locales estructurales y sensitivas basadas en dependencias.

Sea P una propiedad, $M = (A, D)$ un modelo de dependencias entre actores y $limit_P$: $D \rightarrow [1, ||D||]$, una función, una métrica local estructural basada en dependencias para P sobre M es de la forma:

$$P_M: D \rightarrow [0, 1] \text{ tal que } \mathrm{P}_M(d) = \mathrm{P}_{M,D}(d)$$

Definimos las métricas locales sensitivas basadas en dependencias fijando P como:

$$PMax_M = \max d \in D: \mathrm{P}_M(d)$$

4. Ejemplo: Estudio de la conveniencia de usar un sistema software para organizar reuniones

Las métricas estructurales sobre modelos de dependencias entre actores pueden aplicarse en diferentes contextos. Tomando como ejemplo un sistema organizador de reuniones, el análisis se puede realizar a dos niveles. En primer lugar para decidir si un sistema organizador de reuniones software es más conveniente que un sistema humano y, en segundo lugar y en caso que se decida usar un sistema software, para seleccionar la arquitectura de componentes software más adecuada a las necesidades de la organización. En los dos casos las métricas se definirán en términos del tipo de elemento que consideran (basadas en actores o basadas en dependencias), si son globales o locales, y en caso de ser locales se definirá si son sensitivas o no. Asimismo también se define el nivel de precisión que se quiere obtener con la métrica: si se quieren tener en cuenta las características específicas de cada elemento o ya es suficiente con su tipo y si se usan o no los atributos de los elementos.

Supongamos una organización que organiza sus reuniones sin la ayuda de ningún sistema software y se quiere analizar la conveniencia de instalar uno. El modelado del sistema se podría realizar mediante el lenguaje *i** [2], aplicando alguna de las metodologías existentes [2, 3, 10, 11]. Las propiedades que tendríamos en cuenta en dicho análisis serían aquellas que la organización considera más críticas para el proceso, por ejemplo que el sistema resultante respete la privacidad de los usuarios, sea preciso, eficiente y tolerante a fallos. Las propiedades que se definen para estos requisitos son: privacidad de los datos, precisión de los datos, eficiencia en la transmisión de los datos y disolución de la responsabilidad. En las tres primeras propiedades el paso de datos es un factor clave y, por lo tanto, se definen métricas globales y basadas en las dependencias. En cambio, la disolución de la responsabilidad requiere una métrica local sensitiva y basada en actores.

Por razones de espacio no detallamos el estudio completo, pero en la evaluación de las métricas definidas se obtiene que el sistema software es una mejor solución al ser mejor en términos de privacidad y fiabilidad de los datos. La disolución de responsabilidad es mayor en un sistema humano, pues en caso de caída del sistema todo el proceso queda bloqueado hasta que se arregla el problema. En términos de eficiencia los dos tipos de sistema dan resultados parecidos.

Podemos asumir que, con estos resultados, la empresa decide adquirir un sistema software para organizar sus reuniones. Una vez tomada esta decisión, un estudio más profundo debe llevarse a cabo para definir métricas de predicción de costes. Por ejemplo se podría definir una noción equivalente al punto de función [12] para hacer una predicción en términos de las funcionalidades que pueden implementarse, pero este análisis también queda fuera del alcance de este artículo.

5. Ejemplo: Selección de una combinación de COTS para el sistema organizador de reuniones

Para llevar a cabo el proceso de selección, aumentamos la visión del ejemplo del sistema organizador de reuniones para añadir los componentes necesarios en un

sistema de este tipo. En este caso hemos adaptado el modelo propuesto en [2], aprovechando las funcionalidades actualmente disponibles en el mercado para la obtención una solución más avanzada que incluye el control antivirus. También dividimos el organizador de reuniones en dos actores, el organizador y un servicio de entrega de mensajes que será el encargado de enviar y recibir los mensajes de los actores humanos. Además incluimos un administrador del sistema. El modelo *i** resultante se muestra en la figura 1, con algunas dependencias modificadas para que el ejemplo sea más ilustrativo.

En este punto decidimos cual es la mejor arquitectura para el sistema, incluyendo todos los componentes software identificados. El objetivo es obtener arquitecturas como instancias del modelo *i** del sistema y compararlos en términos de ciertas propiedades arquitectónicas.

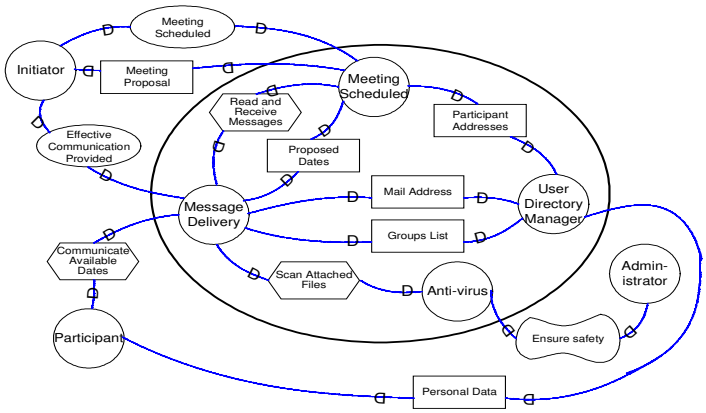

Fig. 1. Fragmento de modelo de actores y dependencias para el ejemplo del sistema socio-técnico de organización de reuniones.

5.1. Alternativas basadas en COTS consideradas

Siguiendo las tendencias actuales, escogemos una solución basada en COTS para nuestro sistema [13]. Muchos componentes COTS existentes en el mercado pueden jugar los roles requeridos en este modelo. La mayoría de organizadores de reuniones incorporan la funcionalidad de correo para permitir a los participantes comunicarse mientras que otros ofrecen requieren el uso de los componentes de correo convencional. De la misma manera, algunos componentes de correo incorporan funcionalidades específicas para gestionar calendarios y agendas (que pueden ser usados para organizar reuniones), así como mecanismos para la protección antivirus y servicio de directorios. Por lo tanto, un proceso de selección fiable debe explorar todos los sistemas posibles formados por combinaciones de componentes COTS existentes pues, en general, todas estas funcionalidades pueden ser cubiertas por uno o varios componentes independientes existentes en el mercado. De esta manera, un organizador de reuniones es implementado como la composición de varios componentes COTS que juegan los roles presentados en la figura 1.

En la figura 2 damos una vista parcial de tres modelos de actores y dependencias para tres arquitecturas de COTS posibles según la asignación de los componentes a los actores (por razones de espacio, sólo mostramos los actores software y el numero de dependencias entre ellos). La arquitectura de la izquierda contiene 4 componentes COTS, cada uno asignado a un único actor. La arquitectura del medio incluye un planificador de reuniones que tiene funcionalidades de gestión de listas de correo y por eso el actor correspondiendo al servicio de directorio esta duplicado. El planificador de reuniones del modelo de la derecha, cumple todas las funcionalidades para la gestión del correo.

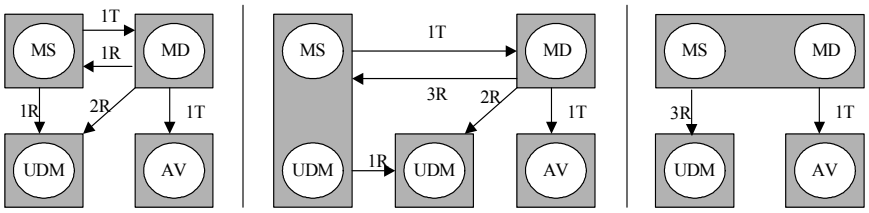

Fig. 2. Tres arquitecturas basadas en COTS diferentes para el sistema organizador de conferencias (MS, organizador de reuniones; MD, envío de mensajes; AV: antivirus; UDM, gestor de directorio; G, objetivo; S: requerimiento no-funcional; R: recurso; T: tarea).

5.2. Evaluación de arquitecturas basadas en COTS

En la comparación de las alternativas exploradas, son útiles tanto las pruebas de verificación de los componentes COTS individuales, como las pruebas de comportamiento de las combinaciones de componentes COTS candidatas. Algunas metodologías como ATAM [14] realizan este análisis arquitectónico, pero se centran sobre todo en las capas físicas de la arquitectura. En [15], proponemos un trabajo preliminar e informal para el uso de métricas estructurales que es la motivación para la formalización propuesta aquí.

Como ejemplo, definimos una métrica estructural para la complejidad de la interfaz de usuario, que mide la propiedad no funcional de usabilidad del sistema [16]. La primera opción es una métrica basada en actores que cuenta aquellos componentes COTS que interaccionan con actores humanos, o sea, que son *dependers* o *dependees* con algún actor humano (parte superior izquierda de la figura 3). Esta primera aproximación es fácil de definir y calcular pero que no es muy precisa, pues suele dar como mejores resultados aquellas soluciones en que un mismo COTS cubre varios roles (coarse-grained COTS-based solution).

Un refinamiento de esta primera métrica consiste en descartar aquellos actores software cuyas dependencias con los actores humanos son exclusivamente no funcionales (*softgoals*), pues este tipo de dependencias no afecta la interfaz con los usuarios. De esta manera, definimos un segundo tipo de métrica basada en dependencias que sólo tiene en cuenta las dependencias de objetivo, de recurso y de tarea. También asignaremos más peso a las dependencias de objetivo que al resto, pues un objetivo suele implicar una descomposición en otras dependencias. Del mismo modo, priorizaremos los tipos de usuario dando menos importancia en

términos de usabilidad al usuario administrador. Esta solución puede verse en la parte superior derecha de la figura 3.

Por último, exploramos una métrica más precisa que asigna pesos individuales a las dependencias que conectan actores humanos y actores software (las otras tendrán un valor nulo). Estas dependencias son: MS, Meeting Scheduled (goal); CAD, Communicate Available Dates (task); PD, Personal Data (resource); MP, Meeting Proposal (resource); ECP, Effective Communication Provided (goal); ES, Ensure Safety (soft-goal). Esta decisión puede tomarse usando criterios cuantitativos o cualitativos. Un criterio cuantitativo prioriza las dependencias de recursos considerando el número de campos que se tienen que rellenar y, las dependencias de tarea, mediante el conteo de *clicks* que se tienen que realizar en el peor de los casos. Para los objetivos y los recursos se puede realizar algún tipo de estimación. En todos los casos se normalizan los resultados para obtener valores entre 0 y 1 (ver parte inferior izquierda de la figura 3).

Para un análisis cualitativo, se pueden aplicar criterios como AHP [17] o *laddering* [18] para ponderar las dependencias entre actores. En la parte inferior derecha de la figura 3 mostramos un ejemplo de priorización de las dependencias entre los humanos y los componentes software usando AHP para definir la función de ponderación.

Para completar las dos últimas métricas hay que definir los valores $g_{A,D}(a)$ teniendo en cuenta las asignaciones de los actores a los componentes COTS concretos. No se incluyen los resultados de esta evaluación por motivos de espacio y porque, para obtener datos validos, hay que considerar el modelo entero y no sólo el fragmento presentado en la figura 1.

$$f_A(a) = \begin{cases} 1, & sort_A(a) = \{software\} \\ 0, & \text{otherwise} \end{cases}$$

$$f_D(d) = \begin{cases} 1, & sort_D(d) = \{goal\} \\ 0.5, & sort_D(d) = \{resource\} \vee sort_D(d) = \{task\} \\ 0, & sort_D(d) = \{soft\ goal\} \end{cases}$$

$$g_{A,D}(a) = \begin{cases} 1, & \exists h \in A: sort_A(h) = \{human\}: \\ & ((a, h, x) \in D \vee (h, a, x) \in D) \\ 0, & \text{otherwise} \end{cases}$$

$$g_D((a, b, x)) = \begin{cases} 0, & sort_A(a) = \{software\} \wedge sort_A(b) = \{software\} \\ 0.5, & a = Administrator \wedge sort_A(b) = \{software\} \\ 0.5, & b = Administrator \wedge sort_A(a) = \{software\} \\ 1, & \text{otherwise} \end{cases}$$

	Interface interaction	f_D(d)
CAD	8	0,8
MP	4	0,4
MS	7	0,7
PD	10	1
ECP	5	0,5
ES	2	0,2

	PCA	IUD	LF	AUP	PD	AD	$f_D(d)$
CAD	1	5	1	1	3	7	0,26
MP	1/5	1	1/3	1/5	1	5	0,08
MS	1	3	1	3	7	5	0,32
PD	1	5	1/3	1	5	5	0,23
ECP	1/3	1	1/7	1/5	1	5	0,08
ES	1/7	1/5	1/5	1/5	1/5	1	0,03

Fig. 3. Cuatro propuestas diferentes de métricas para medir la propiedad de complejidad de la interfaz de usuario.

En el ejemplo de la complejidad de la interfaz de usuario, observamos que la primera métrica da unos resultados completamente diferentes al resto debido a la poca precisión de su definición. La segunda métrica tiene la tendencia correcta pero con poca diferencia, mientras que la tercera y la cuarta son similares. En este último caso

los valores concretos son diferentes debido a que la función de evaluación de dependencias es diferente. Ciertamente las variaciones en los valores usados para esta función pueden incluso cambiar el orden de las alternativas. Pensamos que la priorización de requisitos existente en otros enfoques tales como WinWin [19] pueden ayudar en la asignación de pesos más correctos en los elementos del modelo.

5.3. Otros modelos a considerar

En el conjunto de métricas anterior, hemos instanciado los roles software de un sistema socio-técnico de organización de reuniones con otros roles software correspondientes a tipos de componentes COTS para estudiar el modelo de dependencias entre actores resultante. Sin embargo otras hay otras instanciaciones posibles sobre el modelo de dependencias entre actores original. Por ejemplo, se puede argumentar que la complejidad de la interfaz de usuario dependen más de la variabilidad de proveedores de COTS que del numero de COTS en sí. De esta manera, si el organizador de reuniones y el servicio de entrega de mensajes son productos diferentes de un mismo proveedor, probablemente usaran la misma interfaz. Por esta razón, podríamos definir una asignación de actores usando como asignación los fabricantes de COTS en vez de los componentes mismos.

6. Conclusiones y trabajo futuro

En este artículo hemos presentado un marco para la definición de métricas estructurales para modelos de dependencias entre actores. El objeto a medir son las propiedades del modelo del sistema, que generalmente representan requisitos no funcionales y organizacionales. El marco presentado consta de tres categorías de métricas con dos formas generales cada uno: basadas en actores y basadas en dependencias. El ejemplo presentado para la propuesta es el de un sistema organizador de reuniones, sobre el cual se pueden aplicar métricas estructurales para decidir la conveniencia de usar software de soporte y, entonces, seleccionar una cierta combinación de componentes COTS para el sistema. Como resultado, y ésta es la mayor contribución del artículo, podemos decir que los modelos de actores y dependencias pueden analizarse de forma sistemática respecto a un conjunto de propiedades sobre el modelo y que, para hacerlo, es suficiente con escoger y adaptar el tipo (o tipos) apropiados de métricas para cada propiedad.

Las características más relevantes de la propuesta son:

- **Expresividad.** Tiene en cuenta roles, tipos y atributos tanto en los actores como en las dependencias; distingue los tipos de análisis en global, local y sensible; permite focalizar en actores o dependencias según el concepto predominante para cada métrica en particular.
- **Precisión.** El marco esta basado en definiciones formales de los modelos de dependencias entre actores.

- **Adaptabilidad al coste.** Las métricas se pueden definir de forma más o menos precisa según el esfuerzo puesto en su definición, llegando al nivel de poder ponderar cada elemento del modelo por separado.
- **Soporte Experto.** Debido a que las métricas se definen mediante la adaptación de las formas generales a las necesidades particulares de cada propiedad, el conocimiento experto aumenta la productividad. Estamos desarrollando herramientas de soporte para crear y mantener este conocimiento experto.
- **Reusabilidad.** Las métricas pueden ser usada en modelos diferentes del mismo tipo de dominio.
- **Generalidad.** El marco propuesto no está ligado a ningún lenguaje o formalismo orientado a objetivos en particular, de manera que es fácilmente generalizable.

En la definición de *i** [2], Yu propone conceptos tales como oportunidad y vulnerabilidad que podrían ser modelados en el marco que proponemos pero, hasta donde sabemos, no hay mucho trabajo relacionado en el área. La propuesta más remarcable está en el método AGORA [20] que proporciona técnicas para estimar la calidad de las especificaciones de requisitos en un entorno orientado a objetos. AGORA pone más énfasis en el análisis del grafo AND/OR resultado de la descomposición que en el tipo de modelos de actores de dependencias. A pesar de que es un método expresivo y preciso, no proporciona ningún tipo de forma general para las métricas de manera que su construcción es menos sistemática y es más difícil definir nuevas métricas o rehusar las existentes.

Finalmente, hemos identificado diferentes maneras de seguir esta línea de investigación. Las siguientes enfatizan la utilidad de nuestra propuesta:

- **Construcción de un catalogo de métricas reusables.** Las propiedades de los modelos que hacen referencia a aspectos no funcionales tales como seguridad, eficiencia, ... aparecen a menudo en el análisis de los sistemas y por este motivo, un objetivo futuro es la construcción de un catálogo de métricas para estas propiedades.
- **Integración de este marco con otras propuestas.** En particular estamos interesados en usar este marco en el análisis de arquitecturas de sistemas [6, 14]. Creemos que estas métricas aplicadas sobre los modelos orientados a objetivos proporcionan un primer criterio para clasificar las arquitecturas candidatas.

Agradecimientos

Este trabajo se ha realizado en el marco del proyecto de investigación UPIC, con referencia TIN2004-07461-C02-01, subvencionado por el Ministerio de Ciencia y Tecnología. G. Grau disfruta de una beca de investigación de la Universidad Politècnica de Catalunya.

7. References

[1] A. Dardenne, A. van Lamsweerde, S. Fickas, “Goal-directed Requirements Acquisition”, Science of Computer Programming, 20, 1993, pp. 3-50.

[2] E. Yu. *Modelling Strategic Relationships for Process Reengineering*. PhD. thesis, University of Toronto, 1995.
[3] J. Castro, M. Kolp, J. Mylopoulos, "Towards Requirements-Driven Information System Engineering: The Tropos Project", Information Systems, 27, 2002.
[4] M. Lorenz, J. Kidd. *Object-oriented software metrics: a practical guide*. Prentice-Hall, 1994.
[5] S.R. Chidamber, C.F. Kemerer. "A Metrics Suite for Object-Oriented Design". IEEE TSE 20(6), 1994.
[6] P. Grünbacher, A. Egyed, N. Medvidovic. "Reconciling Software Requirements and Architectures - The CBSP Approach". In *Proceedings of the 5th IEEE International Symposium on Requirements Engineering* (RE'01), 2001. pp. 202-211
[7] E. Yu, "Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering". In *Proceedings of the 3rd IEEE International Symposium in Requirements Engineering* (RE'97), 1997. pp. 226-235.
[8] A. van Lamsweerde, R. Darimont, P. Massonet, "Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learned", In *Proceedings of the IEEE 2nd International Symposium on Requirements Engineering* (RE'95), 1995. pp. 194-203.
[9] E. Yu. "Understanding 'why' in software process modeling, analysis and design". In *Procedings of the 16th International Conference on Software Engineering* (ICSE'94), 1994. pp. 159-168.
[10] Gemma Grau, Xavier Franch, Neil A.M. Maiden. "A Goal Based Round-Trip Method for System Development". To appear in *Proceedings of the Eleventh International Workshop on Requirequirements Engineering: Foundation for Software Quality* (REFSQ'05). June 13-14, 2005. Porto, Portugal.
[11] Gemma Grau, Xavier Franch, Enric Mayol, Claudia Ayala, Carlos Cares, Mariela Haya, Fredy Navarrete, Pere Botella, Carme Quer. "RiSD: A Methodology for Building i* Strategic Dependency Models". To appear in Proceedings of The Seventeenth International Conference on Software Engineering and Knowledge Engineering (SEKE'05). 14-16 July, 2005. Taipei, Taiwan, Republic of China.
[12] D. Garmus, D. Herron. *Measuring The Software Process: A Practical Guide to Functional Measurements*. Prentice-Hall, 1995.
[13] B.C. Meyers, P. Oberndorf. *Managing Software Acquisition*, Addison-Wesley, 2001.
[14] L. Baas, P. Clements, R. Kazman. *Software Architecture in Practice*, *2nd edition.* Addison-Wesley, 2003.
[15] X. Franch, N.A.M. Maiden, "Modeling Component Dependencies to Inform their Selection", In *Proceedings of the 2nd International Conference on COTS-Based Software Systems* (ICCBSS'03), LNCS 2580, Springer-Verlag, 2003.
[16] ISO/IEC Standard 9126-1 Software Engineering – Product Quality – Part 1: Quality Model, 2001.
[17] T.L. Saaty. *The Analytic Hierarchy Process*. McGraw-Hill, 1990.
[18] T.J. Reynolds, J. Gutman. "Laddering Theory, Method, Analysis and Interpretation". Journal of Advertising Research, vol. 28, 1988, pp. 11-31.
[19] B. W. Boehm, P. Grünbacher, R. O. Briggs. "EasyWinWin: A Groupware-Supported Methodology for Requirements Negotiation". In *Proceedings of the 23rd International Conference on Software Engineering* (ICSE 2001), 2001.
[20] H. Kaiya, H. Horai, M. Saeki, "AGORA: Attributed Goal-Oriented Requirements Analysis Method". In *Proceedings of the IEEE 10th International Symposium on Requirements Engineering* (RE), 2002. pp. 13-22.