Una herramienta para asistir la documentación de arquitecturas de software

Juan Marcos Armella, Sánchez Luis Emiliano

Universidad Nacional del Centro de la Provincia de Buenos Aires (UNICEN), Tandil.
juanmarcosarmella@gmail.com, lsanchez@alumnos.exa.unicen.edu.ar

Docente: J. Andres Diaz-Pace

ISISTAN Research Institute CONICET-UNICEN, Tandil.
andres.diazpace@isistan.unicen.edu.ar

Una herramienta para asistir la documentación de arquitecturas de software

Resumen. Durante el desarrollo de software, la arquitectura del sistema y su documentación son factores importantes para el éxito del proyecto, ya que esto permite capturar la estructura de alto nivel del sistema y las decisiones de diseño clave para satisfacer los intereses de los stakeholders del proyecto. Además de servir como repositorio del conocimiento arquitectónico para la organización, el Documento de Arquitecturas, o SAD, debe organizarse en función de sus lectores, es decir, los stakeholders.
Los documentadores generalmente enfrentan distintos desafíos que condicionan sus tareas, tales como: organizar el SAD, satisfacer las necesidades de información de los principales stakeholders (bajo tiempos de entrega ajustados), evitar documentar con demasiado detalle, y mantener el contenido actualizado.
Para tratar estos problemas, en este trabajo se presenta una herramienta que asiste al documentador en la planificación de la documentación de arquitecturas en un marco iterativo e incremental de desarrollo. Se toma como base el método Views & Beyond para la construcción de SADs. Un aspecto novedoso del trabajo es la planificación de tareas de documentación, que se formula como un problema de optimización donde el objetivo es maximizar la satisfacción de los stakeholders sobre el SAD considerando restricciones de costo y consistencia.

1 Introducción

El desarrollo de software es una actividad compleja. Los sistemas tienden a tener mayor tamaño e involucran una gran cantidad de stakeholders, quienes son los interesados en el sistema y son clasificados según sus roles (clientes, desarrolladores, arquitectos, etc.). Los requerimientos y atributos de calidad que demandan los stakeholders del sistema deben ser satisfechos y para esto aparece el término *arquitectura de software* [8] que representa las estructuras de alto nivel del sistema y las principales decisiones de diseño. La documentación de la arquitectura es importante para el éxito del proyecto ya que establece un canal de entendimiento y comunicación para los stakeholders. Esta documentación normalmente se compila en un artefacto denominado Documento de Arquitectura, o SAD (Software Architecture Document). Los stakeholders usualmente poseen distintos intereses respecto a la arquitectura y por ende son proclives a leer secciones y contenidos específicos del SAD para dicha arquitectura.

En la práctica, los documentadores se enfrentan con el problema de cómo estructurar el SAD y generar contenidos "útiles", es decir, que satisfagan las necesidades de información de los stakeholders, pero al mismo tiempo deben evitar documentar demasiado sobre la arquitectura por tratarse de una tarea costosa en cuanto a recursos.

En este trabajo se propone un enfoque para asistir a los documentadores en la generación incremental del SAD, que puede adaptarse fácilmente a procesos iterativos como ADD [6], o Scrum [7], entre otros. Como modelo de referencia, el enfoque toma el método de documentación Views & Beyond [1] del SEI, que propone

adfa, p. 1, 2011.

templates y guías para estructurar el SAD, y particularmente, describe una forma de relacionar los roles de los stakeholders con vistas arquitectónicas predeterminadas (por ej., módulos, componentes y conectores, despliegue). Como soporte para el enfoque, se desarrolló una herramienta prototipo denominada *Arquitectonic*, que consiste en: (i) un *entorno Wiki* para la creación, edición y accesos a secciones de un SAD, y (ii) un *agente de asistencia* responsable de planificar las tareas de documentación, y luego sugerir mejoras para el SAD.

La planificación de tareas representa el principal desafío de la herramienta. El asistente recomienda un *plan de documentación* por iteración, que consiste de una lista de tareas que refleja cuales son las secciones prioritarias del SAD que debieran ser completadas o actualizadas. La construcción de este plan puede verse como un problema de optimización, que se formula como una instancia del *problema de la mochila* (Knapsack Problem). Específicamente, se propone un algoritmo de aproximación para generar planes que maximicen la satisfacción de los stakeholders sobre el SAD actual, sujeto a restricciones de consistencia, para mantener la coherencia de los incrementos del SAD, y de costo, asociado al esfuerzo de trabajo permitido por iteración.

El resto del trabajo se organiza de la siguiente manera. En la sección 2 se presenta el método Views & Beyond. En la sección 3 se describe el enfoque propuesto junto a la herramienta implementada. En la sección 4 se detalla el problema de planificación de tareas y el algoritmo que lo resuelve. En la sección 5 se reportan algunas evaluaciones de optimalidad y eficiencia sobre el algoritmo. Finalmente, en la sección 6 se presentan las conclusiones y propuestas de trabajo futuro.

2 El Método Views & Beyond

El método Views & Beyond [1] (V&B) presenta guías para la creación de un SAD que contemple las necesidades particulares de los stakeholders (o principales consumidores de información del SAD). Según este método, documentar una arquitectura de software involucra documentar las vistas relevantes y la información que las relaciona. La documentación arquitectónica se divide así en dos partes: las vistas (*views*) y la documentación más allá de la vistas (*views & beyond*).

Para la documentación de cada *vista*, V&B cuenta con un template donde cada sección tiene un propósito específico. El template es común para todas las vistas. Para entender las cuestiones más generales del sistema, la organización del documento y las relaciones entre las diferentes vistas, V&B proporciona otro template cuyas secciones comprenden la documentación *más allá de las vistas*. Las secciones que forman a estos dos tipos de templates pueden apreciarse en la Figura 1.

Como es sabido, la arquitectura de un sistema se representa desde múltiples puntos de vista debido a su complejidad natural. Un aspecto interesante abordado por V&B es el de las necesidades específicas de información que cada tipo (rol) de stakeholder posee, como se muestra en la Figura 2. El asistente de *Arquitectonic* presenta dicha relación en una matriz denominada *tabla de preferencias*, que representa el nivel de detalle deseado por cada stakeholder, de acuerdo a su rol, respecto a cada vista de la arquitectura, de acuerdo a su tipo. Por ejemplo, un project manager estará interesado

en la planificación de tareas y la asignación de recursos, por eso su interés por las vistas de asignación de trabajo. Por otro lado, los miembros del equipo de desarrollo necesitarán saber cómo implementar la arquitectura, por eso su interés por las vistas de módulos, y componentes y conectores (C&C) que especifican la estructura, comportamiento y detalles técnicos de los elementos que integrarán el sistema.

Fig. . Templates para *Vistas* y documentación *Mas allá de las vistas*

Template para vistas

1. *Primary presentation*
2. *Element catalog*
   - 2.1. *Elements and their properties*
   - 2.2. *Relations and their properties*
   - 2.3. *Elements interfaces*
   - 2.4. *Behaviour*
3. *Context Diagram*
4. *Variability Guide*
5. *Rationale*

Template para documentación más allá de las vistas

1. *Documentation Roadmap*
2. *How to document a view*
3. *System Overview*
4. *Views*
5. *Mapping between Views*
6. *Mapping to Requirements*
7. *Rationale*
8. *Directory*

Fig. . Tabla de preferencias sugerida por el método View & Beyond (adaptado de [1])

Tipos de vistas arquitectónicas

Necesidad de información de cada stakeholder por vistas de descomposición

Roles

	Module Views					C&C Views	Allocation Views				Other Documentation					
	Decomposition	Uses	Generalization	Layered	Data Model	Various	Deployment	Implementation	Install	Work Assignment	Interface Documentation	Context Diagrams	Mapping Between Views	Variability Guides	Analysis Results	Rationale and Constraints
Project managers	s	s		s			d			d		o				s
Members of development team	d	d	d	d	d	d	s	s	d		d	d	d	d		s
Testers and integrators	d	d	d	d	d	s	s	s	s		d	d	s	d		s
Designers of other systems					s						d	o				
Maintainers	d	d	d	d	d	d	s	s			d	d	d	d		d
Product-line application builders	d	d	s	o	s	s	s	s	s		s	d	s	d		s
Customers							o			o		o			s	
End users						s	s		o						s	
Analysts	d	d	s	d	d	s	d		s		d	d		s	d	s
Infrastructure support personnel	s	s			s	s	d	d	o					s		
New stakeholders	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Current and future architects	d	d	d	d	d	d	d	s	d	s	d	d	d	d	d	d

Key: d = detailed information, s = some details, o = overview information, x = anything

V&B propone un procedimiento de tres pasos para seleccionar las vistas a documentar, e incluir en el SAD. En el primer paso, se construye la tabla de preferencias entre stakeholders y vistas. En segundo lugar, la tabla generada debe analizarse con el fin de reducir la cantidad de vistas a generar. El análisis puede comprender la identificación de las vistas más relevantes del sistema (que son necesarias para una gran cantidad de stakeholders), aquellas vistas menos relevantes

(necesarias para pocos y con poco detalle) y la posibilidad de combinar aquellas vistas que están semánticamente relacionadas (reduciendo su costo de producción). Finalmente, el tercer paso involucra la priorización de los stakeholders y la generación de los contenidos del SAD.

En este punto, el método solo brinda algunas guías y sugerencias de cómo encarar el proceso de documentación para aprovechar los recursos y generar las porciones de documentación claves y que beneficien a los stakeholders importantes. Basado en estas guías, la herramienta *Arquitectonic* desarrollada en el presente trabajo propone automatizar la planificación de las tareas de documentación indicando qué vistas documentar y con qué nivel de detalle.

3 Enfoque y herramienta

El enfoque propuesto, presentado en la Figura 3, se basa en un proceso iterativo e incremental de documentación. En cada iteración se define una estrategia o *plan de documentación* como una lista de tareas para los documentadores. Cada tarea básicamente ataca la redacción de una sección incompleta del SAD actual. Por consiguiente, la lista de tareas permite la elaboración de un subconjunto de las secciones incompletas del SAD. El conjunto de todas las secciones incompletas se conceptualiza como la *pila de tareas* (o backlog). En base a este plan, los documentadores elaboran una nueva versión del SAD, que representa un aumento incremental del artefacto, y que será de utilidad inmediata para los stakeholders del proyecto.

Fig. . Esquema conceptual del enfoque

Como todo proceso iterativo, este comienza con una etapa de inicialización (*configuración inicial*) para preparar la primera iteración de documentación. Esta etapa involucra dos actividades: (i) instanciar el documento SAD y (ii) configurar los parámetros del mismo. El documento SAD se crea a partir de los templates sugeridos por el método V&B, y la configuración de parámetros, aunque no es obligatoria, es necesaria para proporcionar información para la correcta planificación de tareas por parte del asistente. Estos parámetros incluyen: los costos de documentación estimados por secciones, el costo de documentación máximo permitido por iteración, la lista de stakeholders involucrados con sus prioridades y roles, y los tipos de vista asociados a

cada vista arquitectónica del SAD. Con los roles y tipos de vistas, la herramienta puede inferir los intereses de los stakeholders de acuerdo a la tabla de preferencias de la Figura 2. Toda esta información debe ser provista por un experto (por ej., los arquitectos o documentadores), a través de una interfaz gráfica ofrecida por la herramienta.

En cuanto a la planificación de tareas, el asistente podría idealmente definir un único plan de documentación, que comprenda toda la pila de tareas, es decir, sugerir al documentador la redacción de todas las secciones incompletas del SAD. Sin embargo, esta situación se da raramente en la práctica, debido mayormente a restricciones del proyecto. Por esta razón, nuestro enfoque considera dos cuestiones fundamentales para la dinámica del proceso de documentación:

- La documentación se genera en iteraciones, con la idea de satisfacer las necesidades de información en periodos de tiempo relativamente cortos, y acompañando las tareas de diseño arquitectónico. Si toda la documentación se genera una vez que ha concluido el proceso de diseño arquitectónico, no se aporta valor a los stakeholders durante el proceso de diseño. Por eso, se deben definir planes de documentación parciales por iteración, es decir, que comprendan la redacción de un subconjunto de todas las secciones faltantes de acuerdo a la disponibilidad de recursos.
- A medida que el diseño de la arquitectura se va estabilizando, o se incluyen nuevos requerimientos que implican nuevas decisiones de diseño (principalmente en entornos ágiles de desarrollo), nuevas vistas pueden incorporarse para describir la arquitectura, otras pueden combinarse o dividirse para enfocarse en distintas estructuras, etc. Esto implica cambios en la estructura del documento y nuevas secciones a documentar o actualizar.

La herramienta desarrollada consiste en un entorno Wiki integrado con un asistente inteligente, encargado de generar los planes de documentación basándose en las preferencias y prioridades de los stakeholders, el estado de las secciones (documentadas o no) y los tiempos de documentación estimados. Aparte de estos planes, el asistente ofrece sugerencias sobre la arquitectura como la incorporación, combinación o eliminación de vistas irrelevantes, de acuerdo a las preferencias de los interesados. La Wiki, por su parte, facilita la edición colaborativa y consulta concurrente de los artefactos, ofrece plantillas para instanciar documentos de arquitecturas fácilmente, e implementa otra serie de características que facilitan la documentación de arquitecturas de software [2]. En la Figura 4 se muestra una captura de pantalla de *Arquitectonic*.

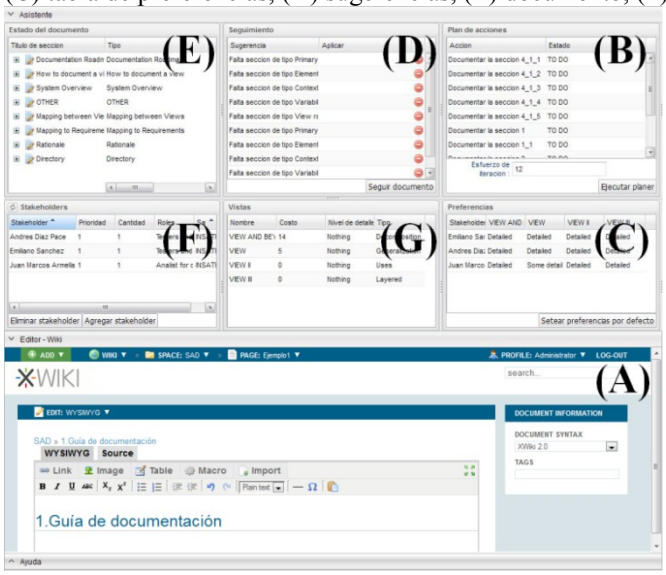
El asistente fue implementado siguiendo un patrón Cliente-Servidor. Como tecnologías del lado del cliente, se combinaron los frameworks GWT[1] y SmartGWT[2]

1 http://developers.google.com/web-toolkit

2 http://www.smartclient.com/product/smartgwt.jsp

para implementar páginas web dinámicas. Del lado del servidor se utilizó MySQL[3] como motor de base de datos y Java Servlets (Java Enterprise Edition)[4] para implementar la funcionalidad. Como entorno Wiki se uso el software XWiki[5] por las características que ofrece: uso de plantillas para instanciar documentos, monitoreo de la actividad de usuarios, y licencia libre, entre otros.

Fig. . Captura de pantalla de la herramienta: (A) wiki, (B) plan de documentación, (C) tabla de preferencias, (D) sugerencias, (E) documento, (F) stakeholders, (G) vistas



4 Planificación de tareas

Como se mencionó anteriormente, el objetivo de la planificación de tareas es seleccionar un subconjunto de secciones sin documentar (del SAD) que maximice la satisfacción de las necesidades de información de la mayoría o, al menos, los stakeholders más importantes del proyecto, considerando la restricción de costo dada por el esfuerzo de trabajo disponible durante la iteración.

Las necesidades de información de los stakeholders se modelan mediante una tabla de preferencias (ver Tabla 1), como la sugerida por V&B, en donde se enlistan

3 http://www.mysql.com

4 http://www.oracle.com/technetwork/java/javaee

5 http://www.xwiki.org

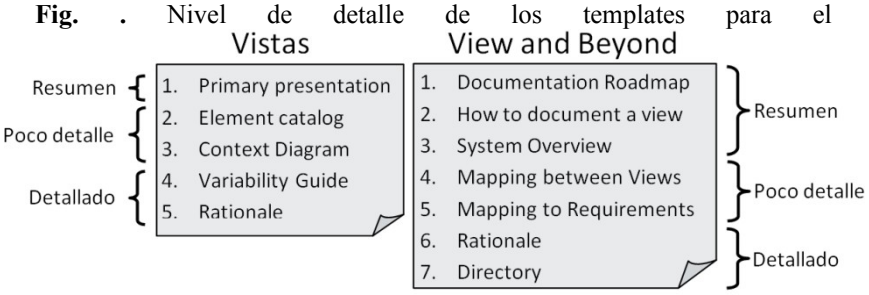
verticalmente los stakeholders del proyecto y horizontalmente las vistas de la arquitectura. Notar que la documentación más allá de las vistas es representada por una columna adicional y modelada por el asistente como si se tratase de otra vista: la vista *Beyond*. Cada intersección representa la cantidad de información o nivel de detalle que requiere cada stakeholder por cada vista, a saber: *Sin interés (-)*, *Resumen* (R), *Poco detalle* (P) y *Detallado* (D). Estas últimas tres representan preferencias.

Tabla . Ejemplo de tabla de preferencias, incluyendo prioridades de stakeholders

Stakeholders (prioridad [0…1])	**Beyond**	**Vista 1**	**…**	**Vista N**
S_1 (P_1)	R	P	…	-
S_2 (P_2)	R	-	…	P
…	…	…	…	…
S_m (P_m)	D	R	…	R

Se considera que se alcanza determinado nivel de detalle de una vista, cuando se completan secciones específicas de la misma. De este modo, documentar secciones es requisito para actualizar el nivel de detalle de las vistas y, así, satisfacer las preferencias de los stakeholders. Siguiendo los templates de V&B para el SAD, el nivel de detalle que requieren los stakeholders se modela tomando los conjuntos de secciones mostrados en la Figura 5. Para alcanzar el nivel *Detallado* se debe además satisfacer el nivel *Poco detalle*, y para alcanzar este último se debe contar con el nivel *Resumen.* Es decir, existe una cadena de relaciones de precedencia: *Detallado* → *Poco detalle* → *Resumen*. Esta restricción de consistencia hace que se deban documentar en orden las secciones dentro de cada vista.

Fig. . Nivel de detalle de los templates para el SAD



Cada preferencia de la Tabla 1 tiene un valor asociado, que se calcula mediante la prioridad del stakeholder dividido por su cantidad de preferencias. La prioridad de un stakeholder es un valor real entre 0 (baja prioridad) y 1 (máxima prioridad) para indicar su grado de importancia. Así, la satisfacción alcanzada por un plan de documentación se evalúa como la sumatoria de los valores de las preferencias satisfechas, ponderada en base a la prioridad de sus stakeholders.

8.1 Formalización del problema

El problema de planificación de tareas se define como un caso particular del problema de la mochila (Knapsack problem). En este problema se tiene un conjunto

de *n* objetos o ítems, cada uno con un *peso* y *beneficio* específicos, y una "mochila" incapaz de soportar más de un peso determinado. El objetivo es maximizar el beneficio total de los items colocados en la mochila sin exceder el peso máximo permitido por la misma. En nuestro escenario de documentación, la mochila es la iteración, cuyo peso máximo es el costo permitido de documentación para dicha iteración. Los ítems son los grupos de secciones a documentar para alcanzar los niveles *Resumen*, *Poco detalle* y *Detallado* presentados en la Figura 5. Es decir, por cada template *i* instanciado en el SAD, ya sea para la documentación Beyond o para cada vista arquitectónica, se define un conjunto de 3 items: R_i, P_i y D_i.

El peso de los ítems es la suma de los costos de documentación de las secciones sin documentar que lo integran, expresado, por ejemplo, en horas-hombre, pues las secciones documentadas se consideran de costo cero. Por otro lado, el beneficio de cada item se define como la sumatoria de los valores de las preferencias con ese nivel de detalle, de acuerdo a lo expresado en la tabla de preferencias. Por ejemplo, para la *Vista 1* de la tabla 1, el ítem *D* tendrá beneficio cero, pues ningún stakeholder tiene ese interés por la vista, mientras que para el ítem *R* de *Beyond* su beneficio es la suma de los valores de las dos preferencias R de la segunda columna.

Una solución al problema consiste en seleccionar un subconjunto de ítems, sujeto a la restricción de peso dada por el costo de iteración. Las secciones incompletas que integran estos ítems representan el plan de documentación. La solución óptima es aquella que maximiza la suma de los beneficios de los ítems seleccionados. Como el beneficio se calcula en base a las preferencias, y esto en base a la prioridad de los stakeholders, la solución optima tiende a favorecer a los principales stakeholders.

8.2 Algoritmo

Como el problema de la mochila es NP-completo, se desarrolló un algoritmo de aproximación con complejidad polinomial para encontrar una solución sub-óptima eficientemente. La solución propuesta se basa en el algoritmo Greedy definido por Dantzig [3]. Este algoritmo consiste en introducir los ítems en la mochila de acuerdo a su orden decreciente de *utilidad*. En una primera etapa, se calcula la utilidad de los ítems mediante la razón *beneficio/peso* de los mismos y se los ordena de forma decreciente en una lista. En una segunda etapa, se adicionan ítems a la "mochila" siguiendo este orden hasta que, por motivo de capacidad, no sea posible seguir introduciendo ítems consecutivos, por lo que se intenta con el siguiente ítem y así sucesivamente hasta completar la capacidad o haber recorrido toda la lista ordenada de ítems.

Este sencillo esquema se debe adaptar para considerar la restricción de consistencia explicada anteriormente, es decir, para todo conjunto (vista) *i*, su ítem R_i debe ubicarse antes de P_i en la lista ordenada, y este antes de D_i. Se usa una cola de prioridad para ordenar los ítems decrecientemente por *utilidad*. Como primera etapa se introducen todos los ítems de nivel *Resumen* (R_i) de cada vista. Por cada ítem que se extrae, se lo incorpora a la "mochila" (que lo acepta o no, si cuenta con la capacidad suficiente) y se agrega a la cola de prioridad el ítem de la misma vista pero de nivel siguiente (P_i o D_i). De esta forma se mantiene la restricción de consistencia

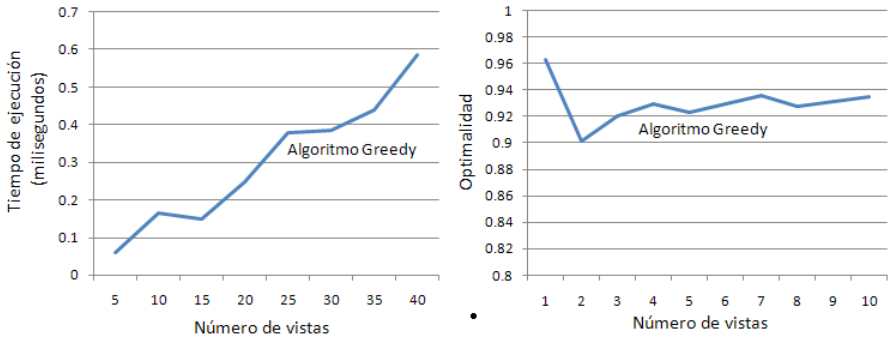
mientras se priorizan los ítems de acuerdo a su utilidad. El procedimiento continúa hasta vaciar la cola de prioridad. El algoritmo, cuyo pseudo-código es presentado a continuación, tiene complejidad *O(n.log(n))*, siendo n la cantidad de conjuntos de ítems (vistas).

Input: *n* conjuntos de 3 items cada uno $\{R_i;P_i;D_i\}$.
Output: Mochila *M* de capacidad *w*.
1: para cada conjunto *i*:
2: agregar R_i a *C*. //cola de prioridad según utilidad
3: mientras *C* no este vacía:
4: *I* ← extraer de *C*.
5: agregar *I* a *M*. //*M* lo acepta o no dependiendo de *w*
6: agregar siguiente item de *I* a *C*.

9 Evaluación

El algoritmo se evaluó en términos de eficiencia (tiempo de ejecución) y optimalidad (beneficio de la solución sub-óptima versus beneficio de la solución óptima) usando escenarios generados aleatoriamente. El parámetro utilizado como variable independiente de los experimentos es el número de vistas *n* del documento, lo que representa *n+1* conjuntos de ítems (considerando un conjunto adicional para la documentación de la vista *Beyond*). En la práctica, un SAD normalmente tiene entre 15-30 vistas. En la Figura 6, se muestran los resultados obtenidos.

Fig. . Resultados experimentales, incrementando el número de vistas del SAD.



Los escenarios fueron generados considerando 10 stakeholders y asignando aleatoriamente sus prioridades y preferencias por las vistas, y los costos de documentación por sección. El costo de iteración se definió como 1/3 del costo total del documento, es decir que la documentación se completaría en 3 iteraciones. Cada instancia del experimento fue ejecutada por lo menos con 30 muestras y el valor promedio es presentado, con la intensión de reducir las fluctuaciones provocadas por la generación aleatoria. El algoritmo fue implementado en Java y evaluado sobre una PC con procesador AMD Athlon de 1,9 Ghz y 1 GB de RAM.

Puede observarse que la performance del algoritmo es aceptable y escalable respecto al número de vistas. Por otro lado, el grado de optimalidad alcanzado por las

soluciones sub-optimas es superior al 90% y se mantiene estable respecto al número de vistas del documento. Para calcular este valor, la solución óptima se computó a partir de un algoritmo basado en backtracking. Se limitó el experimento a escenarios de hasta 10 vistas por la complejidad exponencial de dicho algoritmo de backtracking.

10 Conclusiones y trabajos futuros

En este trabajo se presentó un enfoque basado en el método Views and Beyond para asistir la documentación de arquitecturas de software en el marco de procesos iterativos e incrementales de diseño y desarrollo. El objetivo es maximizar la satisfacción de las necesidades de información de los stakeholders del proyecto en tiempos de entrega ajustados, considerando prioridades y costos de las tareas de documentación.

La herramienta *Arquitectonic* implementa este enfoque y proporciona un ambiente Wiki para la documentación colaborativa de las secciones del SAD. La planificación de tareas se modela y resuelve como una instancia del problema de la mochila. Con este fin, se aplicó un algoritmo greedy para su resolución y se lo evaluó en términos de eficiencia y optimalidad con escenarios sintéticos generados aleatoriamente.

Actualmente, el trabajo se está continuando como proyecto final de carrera con el objetivo de incorporar nuevas características a la herramienta. Principalmente, se está investigando el monitoreo de la actividad de navegación de los stakeholders sobre las secciones del SAD, para inferir sus necesidades de información de manera semiautomática, utilizando técnicas de perfilado de usuario y procesamiento de lenguaje natural [4][5]. Adicionalmente se evaluará el desempeño de la herramienta en escenarios reales de uso de SADs, y se analizarán otras funciones para estimar la calidad de los planes de documentación considerando información adicional de los perfiles de stakeholders, tiempos de entrega (iteración) y otros parámetros.

11 Referencias

1. Clements, P., et al.: *Documenting Software Architectures: Views and Beyond*, 2nd Edition. Addison-Wesley Professional (2010).
2. Bachmann, F., Merson, P.: *Experience using the web-based tool wiki for architecture documentation*. Technical Report TN-041, CMU-SEI (2005).
3. Dantzig, G.: *Discrete-Variable Extremum Problems*, Operations Research Vol. 5, No. 2, pp. 266–288 (1957).
4. Nicoletti, M., Diaz-Pace, J.A., Schiaffino, S.: *Towards software architecture documents matching stakeholders interests*. In Advances in New Technologies, Interactive Interfaces and Communicability. Volume 7547 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012).
5. Nicoletti, M., Diaz-Pace, J.A., Schiaffino, S.: *Discovering stakeholders' interests in Wiki-based architectural documentation*. XVI Ibero-American Conference on Software Engineering (CibSE 2013), Montevideo, Uruguay (2013).

6. Wojcik, R., Bachmann, F., Bass, L., Clements, P., Merson, P., Nord, R., Wood, W.: Attribute-Driven Design (ADD). Technical report CMU/SEI-2006-TR-023 (2006).
7. Schwaber, K.; Beedle, M.: *Agile software development with Scrum*. Prentice Hall. ISBN 0-13-067634-9 (2002).
8. Bass, L., Clements, P., Kazman, R.: *Software Architecture in Practice (3rd Edition)*. Addison-Wesley Professional (2012)