An approach to Context-Based Recommendation in Software Development

Introducción

La estructura del código fuente en lenguajes OO están interrelacionados

La idea general de un sistema de recomendación es estimar la utilidad de un ítem específico a un determinado usuario

En el área de la ingeniería de software un sistema de recomendación puede definirse como:

- ”a software application that provides information items estimated to be valuable for a software engineering task in a given context” *(Robbilard et al)*

En el proceso de desarrollo de software el programador debe lidiar con estructuras que pueden contener cientos elementos en el código fuente

- Estos componentes están relacionados entre sí

Un enfoque de recomendación basada en el contexto puede ser aplicado a los distintos componentes del código fuente

Estudios relacionados

En el pasado se han propuesto diferentes modelos del contexto del desarrollador

- Kertsen y Murphy propusieron un modelo para representar el contexto asociado a una tarea
- Warr y Robillard desarrollaron un plugin para Eclipse que sugiere elementos del código basado en el contexto actual
- Heinemann y Hummel tienen un enfoque de proponer métodos relevantes para una tarea en actual desarrollo

Otros enfoques usan la información contextual para mejorar la recomendación de componentes reusables o ejemplos de código fuente

- Ye y Fisher desarrollaron una herramienta que sugiere de forma pro activa componentes reusables a programadores Java
- Holmes y Murphy propusieron una herramienta para Eclipse que permite encontrar ejemplos de código que pueden ser relevantes para una tarea
- Thummalapenta y Xie desarrollaron un plugin que sugiere ejemplos de ”como hacer llamadas a ciertos métodos”

Modelo de la base de conocimientos

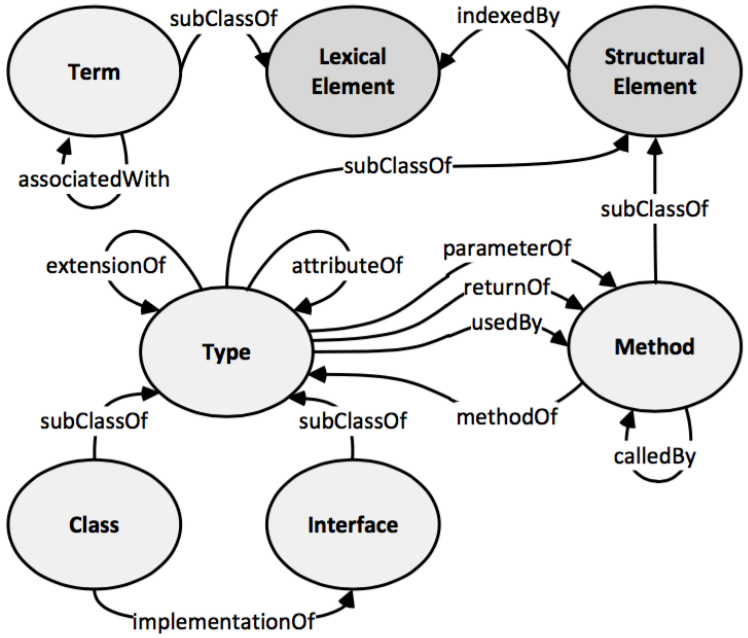


Figure 1: The knowledge base model.

Modelo del contexto

Esta basado en los elementos que se encuentran dentro de código fuente y son más relevantes durante el trabajo del desarrollador

La información del contexto se lo obtiene de forma implícita de las interacciones en el código fuente

La relevancia de cada elemento está dado por un “grado de interés” (DOI)

- Cuando el usuario abre o interactúa con un componente su DOI se incrementa
- Cuando el usuario cierra o deja de interactuar con el componente su DOI se decrementa

Las relaciones que existen entre elementos de código fuente son considerados como “más relevantes” para el programador

- Se calcula igualmente usando el promedio del DOI de los elementos relacionados

Recomendación

Se implementó un método que usa el modelo del contexto para extraer y rankear los elementos relevantes del código fuente

El proceso se basa en el hecho que muchos elementos del código fuente que son necesarios para el desarrollador están relacionados estructuralmente o de forma léxica.

Las recomendaciones obtenidas son entonces rankeadas tomando en cuenta distintos componentes

Retrieval

Las recomendaciones se obtienen dependiendo de dos enfoques:

- Basado en el DOI asignado a cada elemento
- Basado en el tiempo transcurrido desde la ultima vez que un elemento fue consultado

El enfoque basado en el DOI usa la posible relevancia de los elementos del código

- La recomendación incluye un Top N de elementos con el DOI más alto

El enfoque basado en el tiempo se basa en los siguientes hechos:

- Es mucho más probable que los elementos relevantes sean los que fueron accedidos recientemente
- El DOI por otro lado muestra una relevancia a nivel general

Retrieval

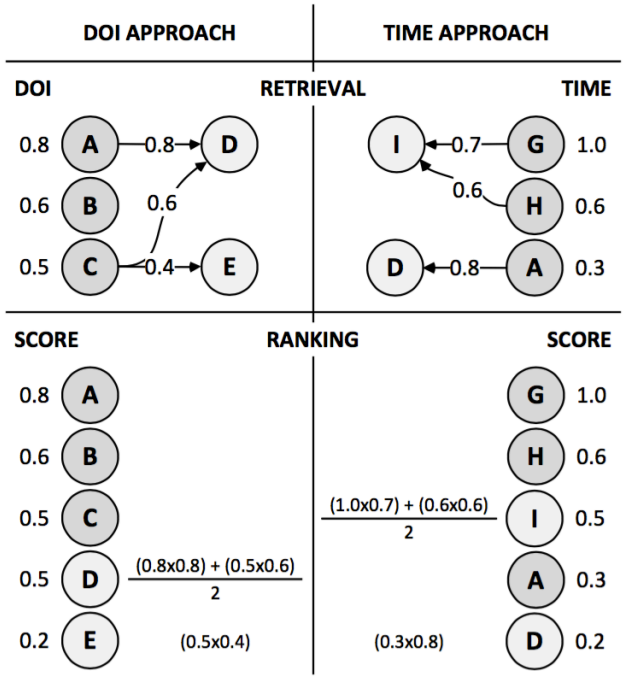


Figure 2: Representation of the retrieval and ranking using the DOI and time based approaches.

Ranking

Las recomendaciones obtenidas son rankeadas dependiendo los siguientes criterios:

- DOI (DOI Score)
- Tiempo
- Relevancia en la estructura
- Relevancia léxica

En el caso del DOI se tiene dos casos

- Cuando el elemento se lo obtuvo de forma directa, su puntaje es el DOI que tiene este
- Elementos relacionados a los anteriores, su DOI “final” esta calculado con su DOI y el DOI asignado a su relación

Para el caso del tiempo se usa un enfoque similar al del DOI

- El tiempo se lo normaliza usando el máximo tiempo entre los elementos en el top N obtenido

Ranking

El puntaje que se le da a la relación estructural esta calculado de la siguiente manera:

- Se define una distancia entre un elemento que esta siendo recomendado con los elementos en el contexto estructural
- La distancia se calcula usando las relaciones estructurales representadas en la base de conocimiento
- La relación entre el costo y DOI es inversamente proporcional
- El costo total que se asigna a un elemento es el costo total que es necesario para llegar de un elemento a otro a través de sus relaciones
- La proximidad de un elemento al contexto está calculado como el promedio del costo para llegar al elemento
- El promedio es normalizado según la siguiente formula

$$1 - \left(\frac{1}{e^x}\right)$$

Ranking

El puntaje léxico está definido como la distancia entre los términos que relacionan los elementos del código fuente que son recomendados con su contexto lógico

- Se usa la relación “associateWith”
- El cálculo es similar al cálculo del puntaje “estructural”
- Se trata de eliminar términos frecuentes en el código como “get, set”

Aprendizaje

Calculo de la influencia de un elemento

$$i_x = rf - rx$$

$$ni_x = 2 \times \left(\frac{i_x - i_{min}}{i_{max} - i_{min}} \right) - 1$$

Calculo de la influencia de forma “balanceada”

$$bi_x^+ = \frac{ni_x^+}{ni_t^+}; \quad ni_t^+ = \sum_{k=0}^{m^+} ni_k^+$$

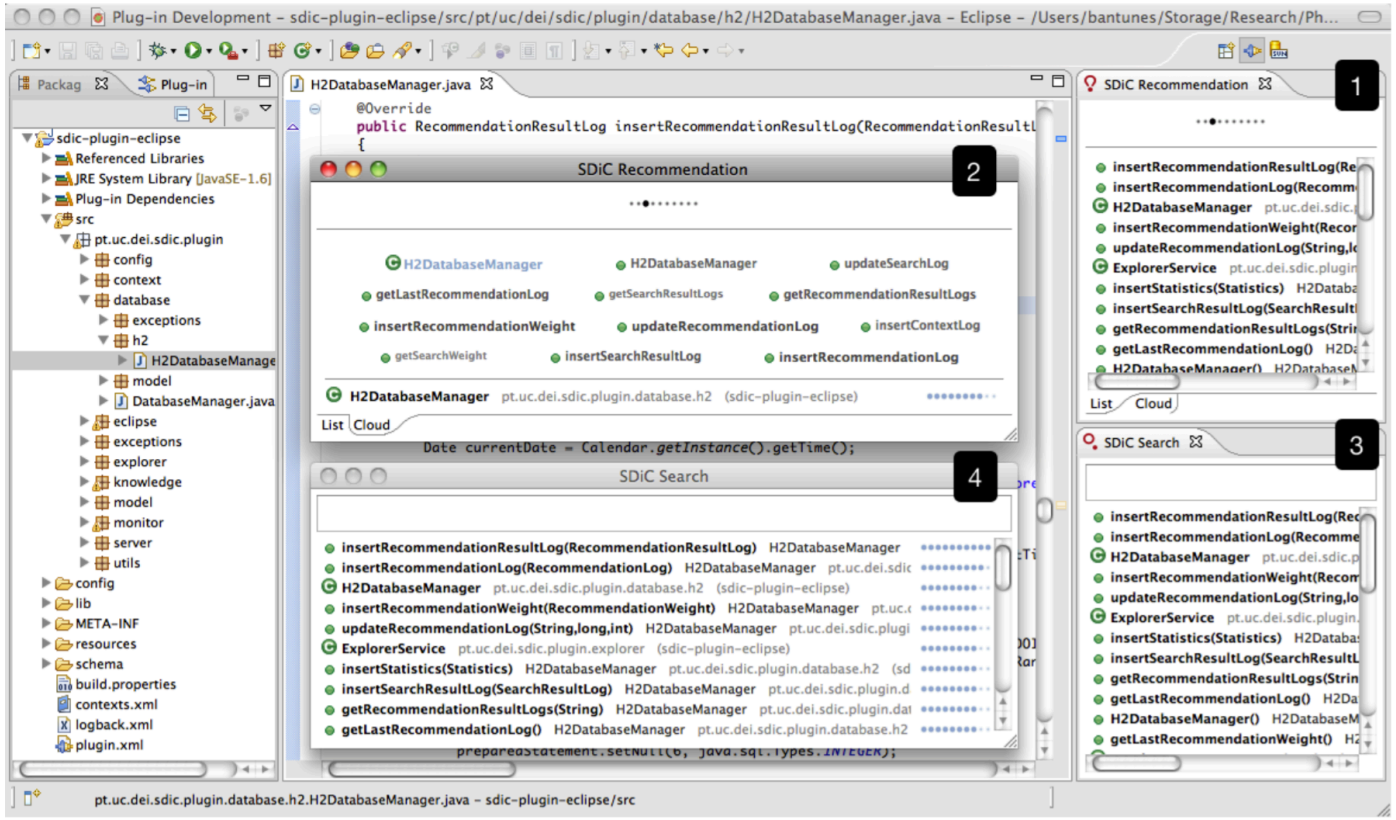
$$bi_x^- = \frac{\left| ni_x^- \right|}{ni_t^-}; \quad ni_t^- = \sum_{k=0}^{m^-} ni_k^-$$

Aprendizaje

Coeficiente de aprendizaje

$$\mu = 0.01 \times \left(1 - \left(\frac{1}{e^{0.2 \times rf}}\right)\right)$$

Prototipo


Plug-in Development – sdic-plugin-eclipse/src/pt/uc/dei/sdic/plugin/database/h2/H2DatabaseManager.java – Eclipse – /Users/bantunes/Storage/Research/Ph...
H2DatabaseManager.java
SDiC Recommendation
1
2
3
4
SDiC Search
List
Cloud
pt.uc.dei.sdic.plugin.database.h2.H2DatabaseManager.java – sdic-plugin-eclipse/src

Evaluación

Se hizo un experimento con un grupo de 5 desarrolladores en la industria y 10 estudiantes

Usaron la herramienta en intervalos de tiempo dentro de un rango de 2 semanas a 3 meses

Resultados cuantitativos

Table 1: Percentage of opened source code elements found in recommendations, and average rankings, per query size.

QS	T	P	F	RD	RT	S	L
1	775/1897	40.9%	6.4	7.6	6.4	7.0	5.5
2	998/1867	53.5%	10.0	10.5	10.9	9.7	8.1
3	985/1807	54.5%	12.6	11.7	14.9	11.4	9.7
4	920/1854	49.6%	14.3	13.8	17.5	12.6	10.4
5	863/1838	47.0%	14.6	13.6	18.7	12.1	10.7
6	763/1842	41.4%	15.3	13.9	19.9	12.1	11.5
7	687/1808	38.0%	14.9	13.5	20.2	11.8	11.5
8	630/1770	35.6%	14.8	14.1	20.3	11.8	11.4
9	637/1845	34.5%	14.7	13.6	20.4	11.2	11.2
10	570/1799	31.7%	15.0	14.3	20.1	11.8	11.0
Total	7828/18327	42.7%	13.1	12.5	16.5	11.1	10.0

Resultados cualitativos

Table 2: Questionnaire results per question.

QUESTION	SCALE	AVG	SD
How would you rate the utility of the recommendation functionality?	Very Low (1) - (5) Very High	3.8	0.93
How would you rate the usability of the recommendation functionality?	Very Poor (1) - (5) Very Good	4.3	1.09
How would you rate the impact of the recommendation functionality in your productivity?	Very Low (1) - (5) Very High	3.2	1.26
How would you rate the overall relevance of recommendations?	Very Irrelevant (1) - (5) Very Relevant	4.0	0.45
How often did relevant recommendations appear among all recommendations?	Very Rarely (1) - (5) Very Often	4.0	1.18
How often did relevant recommendations appear well ranked among all recommendations?	Very Rarely (1) - (5) Very Often	3.6	1.16
How would you rate the improvement in ranking of relevant recommendations over time?	Very Low (1) - (5) Very High	3.8	1.71

Conclusiones

Los resultados obtenidos muestran que usando un enfoque de recomendaciones se puede ayudar de buena manera a un desarrollador

- Esto reduce el esfuerzo que deben invertir los programadores al realizar una tarea

El uso del modelo del contexto fue crucial para identificar los elementos relevantes

- Esto se demuestra por el impacto positivo en las recomendaciones ofrecidas

Por otro lado, se debe trabajar en la interfaz de usuario y mejorarlo

También sería útil analizar los tiempos que se ahorra usando recomendaciones contra otros enfoques como una búsqueda en el código

Referencias

Bruno Antunes, Joel Cordeiro, and Paulo Gomes. 2012. An approach to context-based recommendation in software development. In Proceedings of the sixth ACM conference on Recommender systems (RecSys '12). ACM, New York, NY, USA, 171-178