Transmisi, Vol. 11, No. 1, Juni 2006 : 40 - 44

HIPIKAT: ALAT BANTU PENGEMBANGAN PERANGKAT LUNAK

Eko Handoyo
eko@elektro.ft.undip.ac.id
Jurusan Teknik Elektro, Fakultas Teknik, Universitas Diponegoro

Abstrak

Aspek Sosiologis dan berbagai kesulitan teknis, seperti kurangnya pertemuan informal, dapat mempersulit anggota baru tim pengembang perangkat lunak untuk belajar ke rekan yang lebih berpengalaman. Untuk dapat mengatasi persoalan ini, dikembangkan sebuah alat bantu yang dinamakan Hipikat. Alat bantu ini menyediakan akses memori yang efektif dan efisien dalam pengembangan perangkat lunak. Secara implisit terbentuk oleh semua artifact yang dihasilkan selama pengembangan software tersebut. Dalam tulisan ini dijelaskan pula kegunaan Hipikat dalam menangani modifikasi perangkat lunak. Sehingga banyaknya manfaat yang dapat diambil dalam menangani masalah menunjukan kinerja Hipikat.

Kata Kunci: Software development teams, Project memory, Software artifact.

Saat ini piranti lunak semakin luas dan besar lingkupnya, sehingga tidak bisa lagi dibuat asal-asalan. Pemodelan (modeling) merupakan proses merancang piranti lunak sebelum melakukan pengkodean (coding). Membuat model dari sebuah sistem yang kompleks sangatlah penting karena kita tidak dapat memahami sistem semacam itu secara menyeluruh. Semakin komplek sebuah sistem, semakin penting pula penggunaan teknik pemodelan yang baik. Kesuksesan suatu pemodelan piranti lunak ditentukan oleh tiga unsur, yang kemudian terkenal dengan sebuah segitiga sukses (the triangle for success). Ketiga unsur tersebut adalah metode pemodelan (notation), proses (process) dan alat bantu yang digunakan. Pemahaman terhadap metode pemodelan dan proses disempurnakan dengan penggunaan alat bantu yang tepat, diantaranya Hipikat.

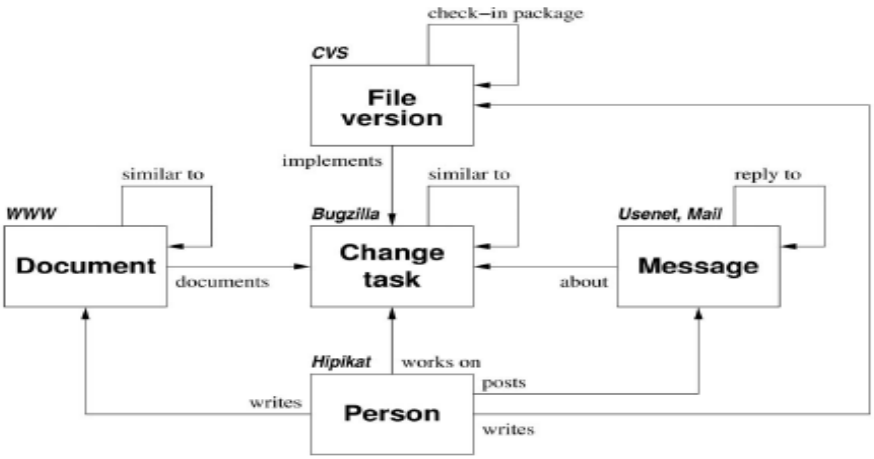
Ada dua fungsi beda yang dibentuk oleh Hipikat. Pertama, alat bantu ini membentuk proyek memori dari artifacts selama merancang pengembangan software. Artifacts tidaklah terbatas pada source program dan dokumentasi, tetapi meliputi komunikasi yang diselenggarakan melalui media elektronik (menyimpan email atau forum diskusi), laporan bug-bug yang ada, dan rencana test. Ke dua, Hipikat merekomendasikan bagi pengembang artifacts memilih proyek memori yang mempunyai relevansi yang sesuai dengan pekerjaanya. Dua fungsi ini diterapkan dalam modul yang beroperasi secara bersamaan dan bebas. Rekomendasi dapat dibuat segera setelah bagian-bagian proyek memori diciptakan. Pembentukan proyek memori merupakan proses yang berkelanjutan: sebagai informasi proyek baru, sumber informasi proyek -tempat penyimpanan kode, database, dan lain-lain Proyek memori terdiri dari proyek artifacts dan juga link antar artifact-artifact yang menandakan hubungan. Jadi kita dapat memodelkan proyek memori sebagai entity relationship diagram. bersamaan dengan pengelompokan artifacts ini sehingga diciptakan perangkat lunak open-source. Ada lima jenis artifacts pada model kita, yaitu change task, file versions, message, document, person. Change task merupakan laporan-laporan yang bermasalah dan diskripsi uraian permintaan yang direkam suatu sistem pelacak seperti Bugzilla. Sumber file versions dicek ke dalam suatu tempat penyimpanan sumber seperti CVS, message dipasang pada forum, majalah dan mailing list, dan dokumen ditempatkan pada proyek web. Person yang menghadirkan pengarang dari artifact. Masing-Masing artifact secara unik diidentifikasi sebagai kunci, hubungan diantara artifact-artifact ini digambarkan pada gambar 1.

Hipikat

Prototipe hipikat telah dirancang dapat beradaptasi dengan mudah dengan proyek manapun yang diikuti dengan model pengembangan software open-source. Hal ini sedikitnya menghasilkan jenis subset artifact yang terdapat di skema memori proyek tersebut. Prototipe hipikat merupakan suatu sistem client-server dan server komunikasi diatas protocol SOAP RPC. Anjuran server dikembalikan dalam format XML. Karakteristik protokol mengijinkan implementasi bahasa dan platform independent. Hal ini mengijinkan suatu klien tertulis pada sebagian proyek tertentu yang sesuai dengan peralatan-peralatan pengembang yang digunakan oleh anggota-anggotanya.

Protocol Hipikat Client Server

Klien meminta ke server untuk rekomendasi dan menampilkan kembali hasilnya kepada pemakai. Klien meminta mengidentifikasi artifact yang terhubung dengan item-item terkait dan mengidentifikasi pemakai tanpa nama. Server memberi jawaban dengan daftar yang cocok dengan klien kemudian membentuk dan menghadirkan format yang mudah dibaca manusia.



Gambar 1. Tipe-tipe artifact dalam proyek memori Hipikat dan hubungan-hubungannya

Masing-Masing item dianjurkan oleh Server dengan diwakili empat nilai, yaitu kunci yang secara unik mengidentifikasi artifact yang digunakan, jika pemakai memutuskan untuk membukanya atau membuat suatu query yang terpasang berurutan; gambaran artifact yang mudah dibaca manusia, alasan merekomendasikan item-item tersebut, kepercayaan diri server atau kekuatan hubungan yang tidak pasti. Nilai kepercayaan diri server dapat digambarkan sebagai pengecekan tingkat tinggi dalam waktu lima menit untuk sebuah versi file yang terhubung ke laporan bug atau numeric. Nilai numeric untuk teks yang memiliki kesamaan karena nilai tidak selalu sebanding dengan kumpulan rekomendasi. Kita memilih secara sederhana menggunakan nilai perhitungan dengan algoritma yang mempunyai kesamaan dengan harapan pemakai hipikat.

Server Hipikat

Server mengimplementasikan tiga fungsi yaitu:

Tempat penyimpanan artifact yang selalu update. Proyek tersebut harus dimonitor untuk penambahan atau perubahan hasil dari sistem pengembang dan evolusinya. Sehingga proyek memori harus update sesuai dengan gambaran penambahan dan perubahan.

Pengidentifikasian mata rantai. Ketika artifacts ditambahkan ke proyek memori, mata rantai yang terkait artifacts harus dikenali dan ditambahkan ke

memori. Penambahan ini mungkin menyebabkan perubahan atau penghapusan mata rantai yang ada untuk beberapa tipe hubungan yang ada.

Pemilihan yang dianjurkan. Hal ini direspon ke query client, artifact yang relevan harus dipilih untuk rekomendasi dan mengembalikan ke pemanggil.

Seperti gambar 2, masing-masing fungsi dibungkus dalam suatu model. Masing-masing model dibagi ke submodel yang memegang satu tipe artifact. Modul tersebut tidak berkomunikasi langsung dengan yang lain tetapi berbagi akses ke database dimana artifact dan hubungan artifact tersebut di simpan. Server ditulis dengan bahasa java. Proyek memori disimpan pada database MySQL.

Database Artifact

Database artifact menyimpan secara primer metadata baru dan mengubah artifact yang dibutuhkan untuk membangun hubungan diantara artifact-artifact yang ada. Text dari isi artifact dan metadata mungkin diindex, tergantung pada tipe artifact. Index text digunakan untuk mencari dan membuat perbandingan yang sama.

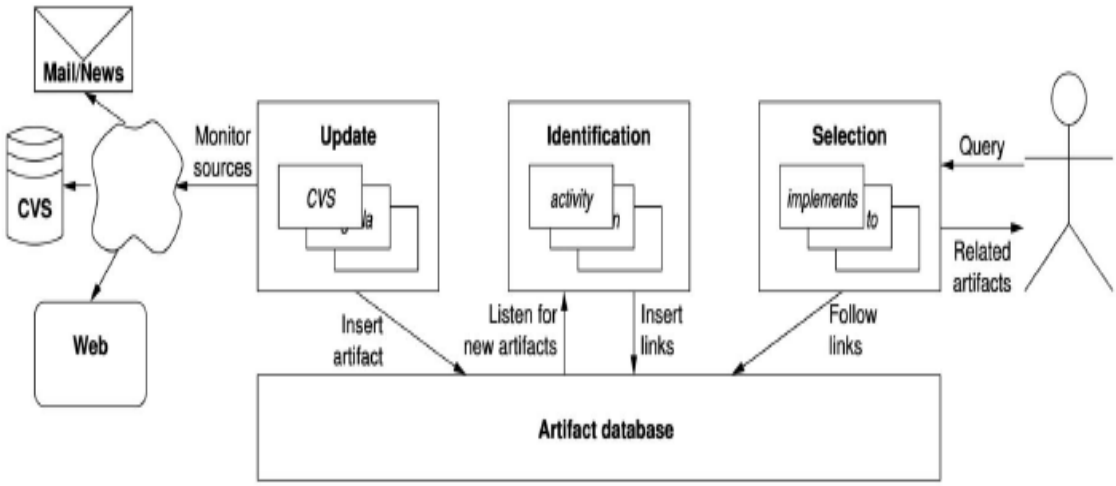
Update

Modul update mempunyai bagian submodul yang memegang masing-masing tipe sumber informasi proyek yang berbeda. Misalnya Bugzilla, CVS atau arsip

mailing list. Masing-masing submodul update memonitor setiap perubahan sumber informasi. Berita dan perubahan artifact dimasukkan ke database artifact hipikat.

Identifikasi

Sistem identifikasi menentukan hubungan diantara hubungan artifact sehingaga datanya dapat tersimpan dalam database. Hubungan tersebut ditentukan dengan menerapkan satu atau lebih heuristics ke artifact yang baru ditambahkan ke database atau modifikasi dengan cara yang sama. Sistem identifikasi dalam Hipikat didesain untuk mendukung heuristic yang banyak. Supervisor Identifikasi mengatur regristasi masing-masing modul heuristics dan antarmuka mereka dengan sitem update.



Gambar 2. Arsitektur server Hipikat

Ada lima contoh modul: log mather, activity matcher, text similarity matcher,CVS chek-in package matcher dan news group thread matcher. Menurut skema dari gambar 1, identifikasi modul juga memberitahukan ketika system update secara berkala mengupdate sumber informasi yang terakhir, dalam hal ini mereka membutuhkan banyak identifikasi proses yang tersimpan misalnya menghitung atau mengecek kembali text yang sama.

Log matcher memanfaatkan ketentuan-ketentuan yang digunakan oleh pengembang open source dengan memasukkan komentar-komentar selama check in versi source code. Hasilnya berisi laporan bug. Activity matcher mencoba melengkapi log matcher dengan membawa faktor keuntungan yang dapat diambil secara alami yang digunakan oleh pengembang. Lebih dari pada itu terlepas pada aturan yang ada. Misal komentar chek in yang ditampilakan tidak teratur. Pendeknya setelah sumber berubah seorang pengembang biasanya akan mengubah status item di bugzilla.

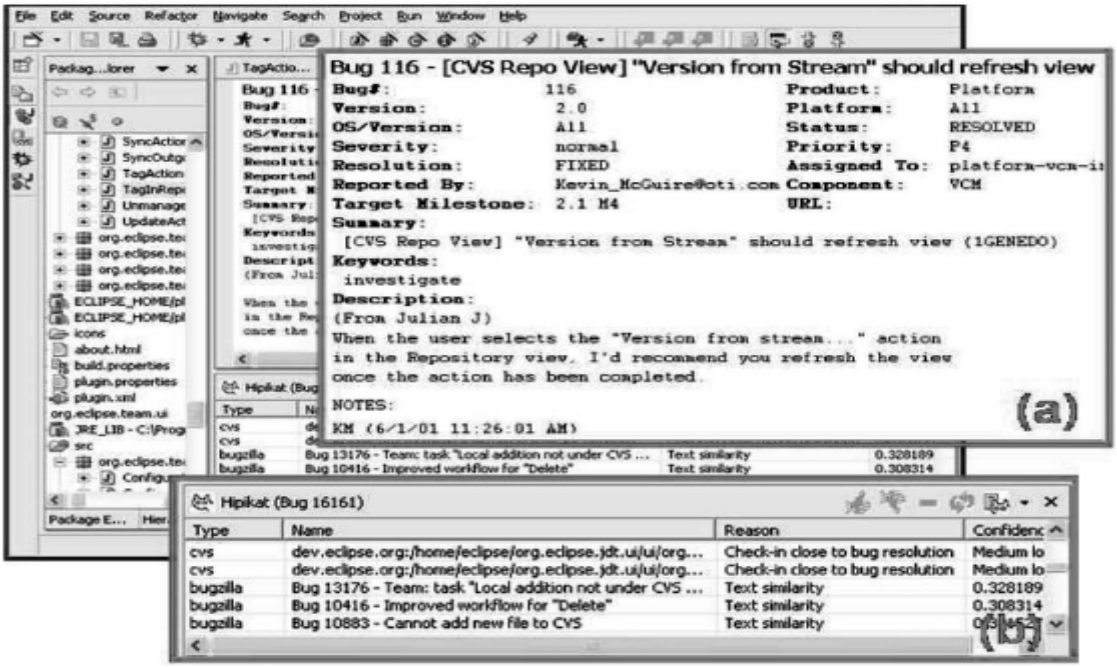
Text similarity matcher, text artifact yang baru diindex, sebagai contoh pendiskripsian Bugzilla dan komentarnya, atau dokumen pada web site proyek. Contoh tersebut berubah menjadi satu dokumen vector. Besaran komponen-komponen tersebut merupakan bobot setiap kata dalam dokumen dan dihitung menggunakan satu produk yang disebut global weight G(i). Local Weight L(i,j). Kita menggunakan kombinasi entropy log untuk dua weight. Local weight dihitung dengan L(i,j)=log(1 + tfij).

Sedangkan global weight dihitung dengan rumus G(i) . Dimana tfij adalah jumlah waktu yang terjadi dalam dokumen. j,N adalah jumlah dokumen dalam kumpulan. , dan di adalah jumlah kumpulan yang diberi istilah i. Text sama yang muncul ini juga digunakan pemakai untuk mencari query,.CVS chek-in package matcher dicek sebagai bagian yang sama ke tempat penyimpanan. News group thread matcher identifikasi modul dicek bersamaan dengan chek in dengan pola yang sama dan disimpan ke tempat penyimpanan. Ini terlihat pada versi lain dengan pengarang dan komentar yang sama dicek dalam enam menit, Thread matcher merupakan idenfikasi sub modul paling sederhana

Pemilihan

Tahap pemilihan membawakan satu set calon yang dianjurkan, menawarkannya dan mungkin menghilangkan item-item dari daftar rekomendasi yang telah dibenahi terlebih dahulu. Pemilihan cara kerja dengan mengikuti alur hubungan permintaan klien (dikhususkan artifact) diharapkan dapat menemukan satu set hubungan artifact-artifact yang ada. Sama dengan identifikasi pemilihan desain untuk mendukung tipe banyak alur hubungan.

Sebagai contoh, satu modul membuat rekomendasi pada CVS dan artifact Bugzilla dengan mengikuti alur-alur hubungan yang dapat diterapkan.Masing-masing modul menyediakan sebab merekomendasikan artifact dan kepercayaan diri menggambarkan kekuatan hubungan yang dapat dibangun. Sebagai contoh laporan sebuah bug dapat dihubungkan ke kedua file yang telah diperbaiki dengan id yang cocok pada daftar log CVS dan aktivitas bug yang cocok. Jika dalam hal ini pemilihan modul hanya menggunakan id log CVS yang cocok, pemilihan modul mempunyai kepercayaan diri yang lebih tinggi.



Gambar 3. Screenshot Hipikat (a) Laporan Masalah (b) Daftar artifact yang berhubungan

Klien Hipikat

Tujuan desain eclipse IDE adalah memudahkan, yang berarti klien dari pada pengaktifan bug yang cocok. Hipikat nampak menyatu dengan IDE dan dapat digunakan dengan peralatan software engineering lain yang terpasang ke Eclipse. sebagai contoh pengembang Eclipse dapat mengakses feature Hipikat search dan Java search yang diikat dengan default distribusi Enclipse. hal ini dapat dilakukan secara sederhana dengan mengklik pada tab appropriate pada dialog search.

Interaksi pemakai dengan Hipikat dijaga sesederhana mungkin supaya mudah dalam memilih artifact dengan membuat query pada project workspace kemudian memilih query hipikat dari menu. Penambahan dan dibuat dari tampilan yang nampak pada IDE, misal daftar file pada workspace, editor java atau tampilan laporan bug (gambar 3a). hal ini dapat dilihat pada tabel 1 untuk daftar keseluruhan dari tipe-tipe artifact dan menempatkannya dalam IDE dimana query Hipikat dapat dibuat. Server merespon dengan daftar hubungan artifact yang dapat dibuka dan digunakan sebagai query. Sehingga pengembang dapat meninggalkan urutan query untuk melihat dan mengakses semua source code dan dokumentasi.

Artifact Hipikat dapat dicari berdasarkan pada tema khusus yang dicari oleh pengembang. Fungsi-fungsi tersebut dapat diakses melalui menu Hipikat search pada dialog search Eclipse. Hasil query atau pencarian ditampilkan pada tampilan hasil Hipikat

dengan eclipse (seperti terlihat pada gambar 3b). Tampilan daftar bagi tpe-tipe yang dianjurkan (halaman web, artikel baru, revisi CVS atau item bugzilla), nama mereka, kenapa hal ini gambar 3 menunjukkan screenshot Hipikat yang sedang jalan dengan Eclipse dianjurkan dan jika diaplikasikan perkiraan kecocokan tertutup.

Klik dua kali pada recommendation akan menghasilkan tampilan artifact yang terbuka seperti terlihat pada gambar 3a. Laporan-laporan bug dan artifact dibuka dengan eclipse; artikel baru dan halaman web dibuka dengan web browser. Klik kanan pada recommendation menghasilkan pop up menu. Dari menu ini pengembang dapat juga membuka item yang dianjurkan supaya dapat dilihat. Dan yang lebih penting dapat digunakan untuk query Hipikat yang lain.

Kesimpulan

Arsip-arsip pengembangan perangkat lunak dapat membantu pembuatan sistem desain memori secara lengkap, dengan membuat keputusan yang cerdik guna mendapatkan cara meyelesaikan permasalahan bug secara tepat. Namun arsip-arsip ini belum dapat disatukan ke kelompok memori meskipun arsip-arsip ini disimpan dan sering diakses menggunakan teknik pencarian informasi. Hipikat sebagai alat yang membentuk group memori secara lengkap guna membantu pendatang baru menyelesaiakan tugasnya secara efektif dan efesien menjadi salah satu solusi yang baik terkait dengan pemecahan masalah dalam mengembangkan perangkat lunak. Hipikat mampu menyediakan pemanfaatan pointer dan kontruksi penyelesaiaan masalah selama mengerjakan tugas. Hipikat dapat juga menganjurkan pemilihan bagian-bagian yang diperlukan secara tepat guna peningkatan pendatang baru dalam menyelesaikan tugasnya.

Daftar Pustaka

Cubranic, D., Murphy, G.C., Singer, J., Booth, K.S., *"Hipikat: A Project Memory for software Development"*, IEEE Transactions on Software Engineering, vol. 31, no.6, Juni 2005, pp 446-465.

Sim, S.E., Holt, R.C., *"The Ramp-Up Problem In Software Projects: A Case Study of How Software Immigrants Naturalize"*, Proc. 20th Int'l Conf. Software Eng, pp. 361-370, 1998.

Herbsleb, J.D., Mockus,A., Finholt, T.A., Grinter, R.E., *"An Empirical Study of Global Software Development: Distance and Speed"*, Proc. 23rd Int'l Conf. Software Eng., pp.81-90, 2001.