MAKALAH

METODOLOGI DESAIN PERANGKAT LUNAK XII

"STORY-DRIVEN MODELLING (SDM)"

Disusun oleh

KELOMPOK 6 :

5200411200 Alfandi Yahya Muhaimin

5200411224 Rizky Ramadhani

5200411228 Sri Uszdevita Syardillah Pohan

5200411232 Rahmita Yida Prihasty

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI & SAINS

UNIVERSITAS TEKNOLOGI YOGYAKARTA

2020

DAFTAR ISI

DAFTAR ISI.....i
BAB I Pendahuluan 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 1
1. Pengertian Story-driven Modelling (SDM) ? 1
1.3 Tujuan.....1
BAB II Pembahasan.....2
BAB III Contoh Metode 5
BAB IV Penutup 7
4.1 (perbandingan dengan metode waterfall, prototype, RAD) 7

BAB I
Pendahuluan

1.1 Latar Belakang

Pada saat ini perkembangan teknologi perangkat lunak telah berkembang pesat dan menjadi pendukung utama bagi sebuah perusahaan. Suatu perusahaan atau lembaga yang menempatkan teknologi perangkat lunak menjadi salah satu pendukung dalam kemajuan perusahaan dapat mencapai rencana strategis organisasi. Perangkat lunak (software) sendiri merupakan program komputer yang terasosiasi dengan dokumentasi perangkat lunak seperti dokumentasi kebutuhan, model desain, dan cara penggunaannya (user manual). Perangkat lunak pada saat ini sudah menjadi kebutuhan khalayak umum di setiap usaha karena dengan memanfaatkan teknologi perangkat lunak akan membantu suatu perusahaan untuk memecahkan sebuah permasalahan yang terjadi.

Story-driven Modelling adalah teknik pemodelan berorientasi objek . Bentuk lain dari pemodelan berorientasi objek fokus pada diagram kelas . Diagram kelas menggambarkan struktur statis dari sebuah program, yaitu blok bangunan dari sebuah program dan bagaimana mereka berhubungan satu sama lain. Diagram kelas juga memodelkan struktur data, tetapi dengan penekanan pada konsep yang agak abstrak seperti tipe dan fitur tipe.

Alih-alih struktur statis abstrak, pemodelan berbasis cerita berfokus pada skenario contoh konkret dan bagaimana langkah-langkah skenario contoh dapat direpresentasikan sebagai diagram objek dan bagaimana diagram objek ini berkembang selama eksekusi skenario.

Dalam artikel ini, akan dijelaskan mengenai apa itu Story-driven Modelling, tujuan, dan yang terlibat dalam Story-driven Modelling

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas maka disimpulkan rumusan masalah :

1. Pengertian Story-driven Modelling (SDM) ?
2. Apa tujuan Story-driven Modelling (SDM) ?
3. Apa saja kegiatan yang terlibat dalam Story-driven Modelling (SDM) ?

1.3 Tujuan

1. Mengetahui apa itu Story-driven Modelling (SDM)
2. Mengetahui tujuan Story-driven Modelling (SDM)
3. Mengetahui apa saja kegiatan yang terlibat dalam Story-driven Modelling (SDM)

BAB II
Pembahasan

A. Pengertian Story-driven Modelling (SDM)

Story-driven Modelling adalah teknik pemodelan berorientasi objek . Bentuk lain dari pemodelan berorientasi objek fokus pada diagram kelas . Diagram kelas menggambarkan struktur statis dari sebuah program, yaitu blok bangunan dari sebuah program dan bagaimana mereka berhubungan satu sama lain. Diagram kelas juga memodelkan struktur data, tetapi dengan penekanan pada konsep yang agak abstrak seperti tipe dan fitur tipe.

Alih-alih struktur statis abstrak, pemodelan berbasis cerita berfokus pada skenario contoh konkret dan bagaimana langkah-langkah skenario contoh dapat direpresentasikan sebagai diagram objek dan bagaimana diagram objek ini berkembang selama eksekusi skenario.

B. Tujuan Story-driven Modelling (SDM)

Tujuan SDM adalah untuk memudahkan analisis, desain, dan implementasi komponen perangkat lunak yang berhubungan dengan struktur data yang dinamis dan sangat kompleks. Meskipun, ini adalah kelas yang sangat penting dari perangkat lunak components, seluruh sistem perangkat lunak mungkin termasuk komponen lain dengan karakteristik yang berbeda, yang harus dikembangkan dengan metode lain. Dengan demikian, SDM tidak dimaksudkan untuk menggantikan pemodelan yang adateknik seperti UML tetapi untuk melengkapinya.

C. Kegiatan yang Terlibat Dalam Story-driven Modelling (SDM)

Berikut ini, akan memberikan gambaran singkat tentang 9 kegiatan yang terlibat dalam SDM, yang akan dibahas lebih rinci sebagai berikut :

- **Fase 1 : Mengumpulkan Persyaratan Informal**

  Langkah pertama dalam SDM adalah memperoleh dan menganalisis persyaratan informalsistem perangkat lunak dengan mendefinisikan apa yang disebut model kasus penggunaan. Ada dua konsep utama dalam model use-case, yaitu aktor dan kasus penggunaan . Dalam hal ini, aktor mewakili peran tertentu yang dapat dimainkan pengguna, sementara penggunakasus didefinisikan sebagai urutan transaksi dalam dialog dengan sistem.

- **Fase 2 : Story Boarding**

  Story Boarding adalah cara yang sangat efektif untuk mengajar pengembang junior, dan mengkomunikasikan visi arsitektur**.** Pada saat ini, penting untuk dicatat bahwa story boarding dirancang sebagai suatu kegiatan dimana pengembang hanya menggambarkan ide bagaimana sistem akan bekerja. Selain itu, papan cerita adalah subjek yang sangat baik untuk diskusi tentang ide-ide yang digariskan antara yang berbeda pengembang.

- **Fase 3 : Menurunkan Struktur Kelas Statis**

  Mengingat sejumlah papan cerita, mudah untuk memperoleh informasi tentang struktur kelas statis dari asisten perangkat lunak. Di sini, kami menggunakan kembali

banyak ide. Pada langkah pertama kami mengidentifikasi kelas untuk objek dengan sifat yang sama. Kemudian kami mendefinisikan hierarki pewarisan dengan mencari kelas dengan properti umum. Selanjutnya, kita dapat menentukan asosiasi dan agregasi dan bahkan beberapa kendala dinamis dari situasi sampel yang ditentukan. Bersama - sama, papan cerita dan diagram kelas satu ini mendefinisikan model domain untuk aplikasi kita.

- **Fase 4 : Merancang Model Konseptual**

  Pada fase ini kami meninjau kembali papan cerita fase 2 dan mengembangkan ide bagaimana sistem akan bekerja. Juga bukan-mally, kita harus mengerjakan ulang struktur objek yang ada terhadap kebutuhan algoritmik dan menambahkan fasilitasing dan struktur data sementara. Ini akan menghasilkan papan cerita yang lebih detail yang memberikan pandangan tentang langkah-langkah perantara dan pada struktur data dari algoritma yang digunakan. Selain itu, seseorang diturunkan diagram kelas yang diperluas dan disempurnakan yang terdiri dari objek tambahan. Akhirnya, kami mengasosiasikan ident-operasi tified ke kelas. Dengan demikian, fase ini menghasilkan model konseptual dari aplikasi yang diinginkan.

- **Fase 5 : Menurunkan Operasi Dinamis**

  Dalam kegiatan ini, ide-ide yang dituangkan dalam storyboard fase-fase sebelumnya dielaborasi. Sekarang kita melihat struktur objek konkret dan algoritma yang mewujudkan sistem yang diinginkan. Dalam SDM ini adalah dilakukan dengan menggunakan apa yang disebut diagram cerita . Diagram cerita adalah adaptasi dan peningkatan pro-sistem penulisan ulang grafik grammed ke notasi UML (diagram aktivitas) dan objek model data berorientasi. Diagram cerita menggunakan notasi grafis tingkat tinggi yang sangat mirip dengan papan cerita. Namun, sementara papan cerita menggambarkan situasi sampel tertentu, diagram cerita menentukan yang umum kasus. Dengan kata lain diagram cerita menentukan sistem umum yang mengimplementasikan skenario sampel dijelaskan dalam papan cerita.

- **Fase 6 : Memvalidasi Model**

  Setelah menyelesaikan spesifikasi struktur kelas dan operasi yang diimplementasikan oleh serangkaian cerita diagram model harus divalidasi. Saat ini, kami menggunakan tata bahasa grafik PROGRES lingkungan rekayasa untuk tujuan ini. Diagram cerita dapat dengan mudah diterjemahkan ke dalam PROGRESaturan transformasi dengan hanya sedikit modifikasi. Namun, kami harus mendefinisikan beberapa tambahanfungsi perpustakaan yang mendukung urutan yang diurutkan dan diurutkan. Kemudian lingkungan PROGRES memungkinkan eksekusi langsung diagram cerita oleh penerjemah terintegrasi. Selanjutnya, lingkungan ini menyediakan browser grafik dengan fasilitas tata letak otomatis yang memungkinkan pengembang untuk tetap melacak efek operasi pada struktur objek.

- **Fase 7 : Verifikasi Model**

  Diagram cerita memiliki semantik formal yang didefinisikan dengan sangat baik yang memungkinkan verifikasi sistem kritis.properti tem. Oleh karena itu, teori

tata bahasa graf memberi kami sarana yang kuat membuat penetapan dan bukti invarian dan bukti pengakhiran dan banyak impor lainnya langkah verifikasi cukup mudah.

- **Fase 8 : Menurunkan dan Memvalidasi Prototipe Standar**

  Setelah model diuji oleh pengembang, model itu juga harus divalidasi oleh pelanggan untuk memperoleh persyaratan lebih lanjut sebelum sistem perangkat lunak akhirnya diimplementasikan dan disesuaikan. Untuk tujuan PROGRES menyediakan sarana untuk menghasilkan prototipe cepat berdasarkan arsitektur standar untuk sistem interaktif. Prototipe yang dihasilkan terdiri dari beberapa komponen tampilan dan menu untuk memanggil operasi antarmuka dari sistem yang dimodelkan

- **Fase 9 : Meraih dan Menyesuaikan Implementasi**

  Jika fungsionalitas prototipe yang dihasilkan memuaskan, langkah terakhir adalah mengimplementasikan perangkat lunak sistem. Dalam tugas ini persyaratan sistem lebih lanjut seperti distribusi, independensi platform, efisiensi, toleransi kesalahan, dll harus dipertimbangkan. Oleh karena itu, biasanya diagram kelas harus disempurnakan dan diperpanjang. Selain itu, kita harus mengembangkan antarmuka pengguna (grafis) yang profesional. Satu kemungkinan Kemampuan untuk mendapatkan implementasi adalah dengan menggunakan kompiler PROGRES untuk menghasilkan kode C atau Modula yang secara semantik setara dengan model yang ditentukan. Namun, PROGRES menghasilkan kode membutuhkan kerangka arsitektur yang agak kaku dan jika misalnya kemandirian platform atau integrasi tanpa batas dengan bahasa berorientasi objek diperlukan, pendekatan ini tidak cukup. Oleh karena itu, kami menyediakan sistem terjemahan tematik dari diagram kelas ke dalam implementasi C++ atau Java langsung dari sistem yang ditentukan.

BAB III
Contoh Metode

Automata waktunya adalah perpanjangan dari automata terbatas oleh satu set real-variabel bernilai C, yang disebut jam. Jam digunakan untuk mengukur kemajuan waktu selama eksekusi otomatis. Waktu terus berkembang secara konstan dan seragam di semua jam robot. Dalam automata berjangka waktu, eksekusi dapat dibatasi dengan menempatkan kondisi tambahan referensi jam ke negara dan transisi. Dengan demikian, perilaku otomat tidak tidak hanya bergantung pada input, tetapi juga pada titik waktu ketika input tersebut diterima. Dalam subbagian berikut, pertama-tama kita akan mendefinisikan automata berwaktu dan jaringan automata berwaktu. tomata seperti yang digunakan oleh pemeriksa model UPPAAL. Setelah itu, kita akan mendefinisikan semantik operasional jaringan automata waktunya.

Definisi

Sebelum kita dapat mendefinisikan automata waktunya, kita harus mendefinisikan batasan jam yaitu digunakan untuk menentukan kondisi pada nilai jam dari otomat waktunya. Di jambatasan didefinisikan sebagai berikut.

Definisi "Kendala Jam"

Misalkan C adalah himpunan jam, c_i, c_j C. Batasan jam adalah rumus konjungtif dari atom kendala berbentuk c_i n atau $c_i - c_j \sim$ n dengan $\{<,\leq,=,\geq,>\}$ dan n N. Batasan jam atomik dapat berupa perbandingan langsung dengan bilangan bulat, misalnya c 10, atau perbandingan perbedaan jam dengan bilangan bulat, misal $c_1 c_2$ 10. Maka jam atom ini kendala dapat digabungkan untuk membangun kondisi jam yang lebih kompleks. Batasan jam itu digunakan sebagai invarian harus dibatasi.

Definisi "Konstrain Jam Invarian"

Batasan jam yang digunakan sebagai invarian dibatasi untuk konjungsi tertutup ke bawah batasan jam atom dalam bentuk c n di mana $\{<,\leq\}$. Menggunakan definisi batasan jam, sekarang kita dapat mendefinisikan otomat berjangka waktu.

Definisi "Otomat Berwaktu"

Mari C menjadi set jam dan B (C) satu set jam kendala menggunakan jam di C. A berjangka waktu otomatis A adalah tupel (N,l_0,E,I) di mana

• N adalah himpunan terbatas dari lokasi (atau node),

• l_0 N adalah lokasi awal,

• E N × B(C) × × 2^C × N adalah himpunan sisi dan

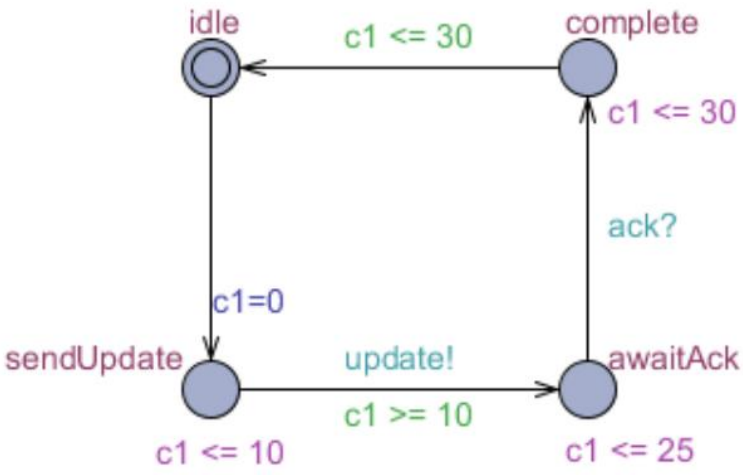
• I : N → B(C) menetapkan invarian ke lokasi.

Kami akan menulis $l, a, \rightarrow l$ ketika (l, , a, , l) E. [BY03]

Seperti otomata terbatas, otomat berjangka waktu terdiri dari satu set status dan transisi. tepat waktu automata, keadaan biasanya dilambangkan sebagai lokasi karena ketergantungan waktu dari kondisi saat ini. Satu lokasi ditandai sebagai lokasi awal. Setiap lokasi dapat membawa invarian yang menyatakan nilai maksimum jam yang mungkin dicapai saat tinggal di ini lokasi. Ketika nilai

jam ini kedaluwarsa, lokasi harus ditinggalkan oleh transisi. Jika ini tidak mungkin, terjadi kebuntuan waktu berhenti. Transisi dari otomat waktunya dapat membawa tindakan input dan output dari alfabet, penjaga waktu dari B(C) dan pembaruan jam. Penjaga waktu membatasi eksekusi transisi ke interval waktu tertentu saat pembaruan jam, juga disebut pengaturan ulang jam, mengatur ulangnilai dari jam ke 0.

Gambar 2.1 menunjukkan contoh otomat berjangka waktu dalam sintaks konkret UPPAAL.



Gambar 2.1

Automaton waktunya memiliki empat lokasi idle, sendUpdate, waitingAck, dan complete whileidle berfungsi sebagai lokasi awal. Semua lokasi kecuali untuk invarian carry idle, misalnya c1 <= 10 untuk status sendUpdate. Transisi idle → sendUpdate membawa pembaruan jam c1=0 yang mengatur ulang jam c1 ke 0. Transisi berikutnya sendUpdate → menungguAck membawa pembaruan tindakan keluaran dan penjaga waktu c1 >= 10.

Automata berjangka waktu memungkinkan komposisi paralel dengan menggunakan opsi komposisi paralel CCS pengatur. Komposisi paralel dari satu set automata berjangka waktu menghasilkan jaringan automata berjangka waktu yang dapat didefinisikan sebagai berikut.

Definisi "Jaringan Automata Jangka Waktu"

Jaringan otomata berwaktu adalah komposisi paralel $A_1|...|A_n$ dari set otomata berwaktu $A_1,...,A_n$, disebut proses, digabungkan menjadi satu sistem dengan komposisi paralel CCS operator dengan semua tindakan eksternal disembunyikan. Komunikasi sinkron antara proses adalah dengan sinkronisasi jabat tangan menggunakan tindakan input dan output; komunikasi asinkron-tion adalah dengan variabel bersama [...]. Untuk memodelkan sinkronisasi jabat tangan, alfabet tindakan diasumsikan terdiri dari simbol untuk tindakan input yang dilambangkan a? dan tindakan keluaran dilambangkan dengan a!, dan tindakan internal diwakili oleh simbol yang berbeda .

BAB IV
Penutup

4.1 (perbandingan dengan metode waterfall, prototype, RAD)

survey modeling	Waterfall	Prototype	RAD
Model pengembangan Survey ini . cocok di gunakan untuk pemodelan berorientasi objek . fokus pada diagram kelas . Diagram kelas menggambarkan struktur statis dari sebuah program.	Model pengembangan Waterfall cocok digunakan untuk sistem atau perangkat lunak yang bersifat generik, artinya sistem dapat diidentifikasi semua kebutuhannya dari awal dengan spesifikasi yang umum serta sesuai untuk tugas akhir/skripsi yang memiliki tujuan untuk membangun sebuah sistem dari awal yang mengumpulkan kebutuhan sistem yang akan dibangun sesuai dengan topik penelitian yang dipilih sampai dengan produk tersebut diuji	Model pengembangan Prototype lebih cocok untuk sistem atau perangkat lunak yang bersifat customize, artinya software yang diciptakan berdasarkan permintaan dan kebutuhan (bahkan situasi atau kondisi) tertentu dan sesuai untuk tugas akhir/skripsi yang memiliki tujuan untuk mengimplementasikan sebuah metode atau algoritma tertentu pada suatu kasus.	Model pengembangan RAD lebih cocok untuk sistem atau perangkat lunak yang bersifat customize, berskala besar dan memerlukan waktu yang lebih singkat artinya software yang diciptakan berdasarkan permintaan dan kebutuhan (bahkan situasi atau kondisi) tertentu dan sesuai untuk perangkat lunak memiliki tujuan untuk mengimplementasikan sebuah metode atau algoritma tertentu pada suatu kasus, serta memiliki kemungkinan untuk kebutuhan pengembangan kembali dalam jangka waktu yang cukup panjang.