JMP : Volume 6 Nomor 2, Desember 2014, hal. 53 - 64

MODEL-DRIVEN DEVELOPMENT : FASE AWAL VERIFIKASI MODEL DESIGN REKAM MEDIS ELEKTRONIS MENGGUNAKAN PERUMUSAN GRAF LENGKAP

Acep Taryana[1], Bangun Wijayanto[2]
Fakultas Teknik, Universitas Jenderal Soedirman
[1]aetthea71@gmail.com, [2]bangun.wijayanto@gmail.com

Naoyasu Ubayashi
Faculty of Information Science and Electrical Engineering, Kyushu University
ubayashi@ait.kyushu-u.ac.jp

Joko Setyono
Fakultas Kedokteran, Universitas Jenderal Soedirman
ab7616sa@yahoo.com

Abstract. *In this paper will be shown a graph formulation as a formal approaches in research Model-Driven Development (MDD) with a case study : the development of Electronic Medical Record (RME) on the scope of the public health center. The model was designed using UML notation and be selected a State Machine diagram that represents prerequisite user needs (requirements). Before the model is derived (driven) into the skeleton code, the accuracy of the state machine must be verified. In order for the State Machine can be verified by formal approach, the State Machine should be first transformed into a propositional formula using the complete graph approach, and partial models. The initial phase of verification will check the suitability of the model with the requirements in Propositional Normal Form (PNF) using SAT Solver, respectively as* $\boldsymbol{\Phi_M}$ *and* $\boldsymbol{\Phi_p}$ *. SAT solver will provide a design decision, whether a requirement represented in the model or not. If these requirements are not hold in the model, the requirement is not certainty (uncertain) and model must be redesigned.*

Keywords: *software engineering, uncertainty, formal method, graph, UML*

Abstrak. *Pada makalah ini akan ditunjukkan perumusan graf sebagai pendekatan formal dalam penelitian Model-Driven Development (MDD) dengan studi kasus pengembangan Rekam Medis Elektronis (RME) pada ruang lingkup Puskesmas. Model dirancang menggunakan notasi grafis Unified Modelling Langguage (UML) dan dipilih diagram State Machine sebagai representasi kebutuhan prasyarat pengguna (requirements). Sebelum model diturunkan (driven) ke dalam kerangka program, State Machine harus diverifikasi ketepatan modelnya sehingga meyakinkan (certainty). Agar State Machine dapat diverifikasi dengan pendekatan formal maka State Machine harus ditransformasikan kedalam formula proposisi dengan bantuan perumusan graf lengkap, dan model parsial. Tahap awal verifikasi akan mengecek kesesuaian model dengan kebutuhan prasyarat dalam Propositional Normal Form (PNF) menggunakan SAT Solver, berturut-turut ditulis* Φ_M *dan* Φ_p*. SAT Solver akan memberikan sebuah keputusan perancangan, apakah sebuah requirement tertuang dalam model atau tidak. Jika requirement tersebut tidak terwakili maka requirement tersebut kurang meyakinkan (uncertain) dan harus dilakukan perancangan ulang model.*

Kata kunci: *rekayasa perangkat lunak, ketidakpastian, metode formal, graf, UML*

1. PENDAHULUAN

Terdapat beberapa permasalahan pengembangan perangkat lunak dimana program dibangun melalui model yang belum final, penyelesaian program diperoleh dengan coba – coba (*trial and error*). Permasalahan ini disebabkan oleh kesulitan untuk mendapatkan model yang pasti sebelum fase pengembangan program. Kadangkala, model yang ditetapkan masih mengandung ketidakpastian. Secara praktis di dunia bisnis, proyek pengembangan perangkat lunak telah berkembang dengan sangat cepat yang didukung dengan berbagai perkakas seperti perkakas perancangan, perkakas pemrograman. Pengembangan model didukung dengan tersedianya berbagai perkakas untuk perancangan model seperti **Rational Rose**, **Visual Paradigm**. Namun, perkakas tersebut belum mampu memeriksa kepastian model yang dibuat. Saat ini, ketidakpastian model merupakan salah satu topik hangat penelitian Model-Driven Development (MDD).

Makalah ini akan membahas tentang penerapan pendekatan *Model-Driven Development* (MDD) pada pengembangan RME. Saat ini MDD merupakan topik hangat penelitian para pakar di bidang *software engineering* (M. Famelis dkk, 2012). Pengembangan aplikasi menggunakan pendekatan MDD memberikan kecepatan validasi, model dapat divalidasi dan diverifikasi menggunakan pendekatan formal secara matematis sehingga kepastian model dapat ditemukan secara eksak. Melalui dukungan perkakas MDD, komunikasi antar programmer dengan perancang pada arah maju (*forward*) maupun arah balik (*reverse*) dapat diselenggarakan dengan tersistem.

2. METODE PENELITIAN

Secara spesifik, makalah ini akan menginvestigasi beberapa literatur yang dapat dijadikan acuan awal untuk membahas verifikasi model menggunakan pendekatan formal seperti graf. Kasus yang digunakan untuk pembahasan tersebut menggunakan hasil penelitian tentang pengembangan purwarupa Rekam Medis Elektronis (RME) yang dikembangkan dari unit kerja layanan kesehatan terkecil yang dilakukan di Puskesmas dan Posyandu (Taryana A dkk, 2013). Pembahasan verifikasi akan diawali dengan membahas tentang konsep MDD, Konstruksi Kebutuhan Prasyarat, Verifikasi, Kesimpulan dan Saran.

3. HASIL DAN PEMBAHASAN

A. Model-Driven Development

Model-driven development (MDD) adalah sebuah paradigm untuk menulis dan mengimplementasikan program komputer secara cepat, efektif dan berbiaya minimum. Pendekatan MDD untuk pengembangan perangkat lunak memungkinkan orang untuk bekerja sama dalam sebuah proyek bahkan dengan tingkat pengalaman masing-masing yang sangat bervariasi. Hal ini memungkinkan perusahaan untuk memaksimalkan kerja yang efektif pada sebuah proyek dan meminimalkan *overhead* yang diperlukan untuk menghasilkan perangkat lunak yang dapat divalidasi oleh pengguna akhir dalam waktu sesingkat mungkin. MDD, merupakan metodologi tangkas (*agile*), terus berkembang untuk memenuhi kebutuhan bisnis.

MDD adalah sebuah ide yang mampu mentransformasikan model ke dalam system komputer. Model dapat berbentuk berbagai macam seperti *Parametrics* for *controllers, control diagrams*, program, UML. Fokus penelitian ini adalah pembahasan tentang pengembangan model dan program menggunakan notasi UML. UML singkatan dari *Unified Modelling Langguage,* merupakan notasi dalam bentuk diagram untuk membangun model pada tahap analisis dan perancangan system (Grady Booch dkk, 1998).

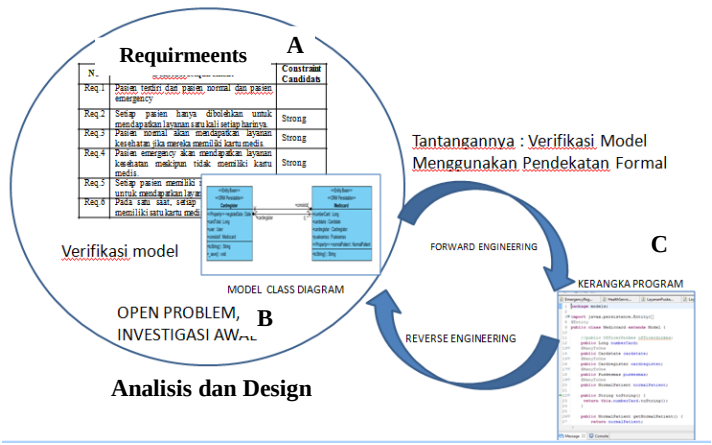
MDD merupakan pemikiran bagaimana membawa pengembangan perangkat lunak sebagai sebuah rangkaian dari transformasi model yang berawal dari kebutuhan prasyarat (*requirements*) menuju sebuah model yang independen dan spesifik, dan kemudian membangkitkan kode sumber (*code*) yang dapat dikompilasi dalam system komputer. Oleh karena itu pengembangannya berorientasi *model-centric* dan banyak aktifitas, meliputi analisis perancangan awal dan pembangkitan kasus uji (*test case*), yang semuanya akan didasarkan pada model menggunakan UML.

Pengembangan system komputer menggunakan pendekatan MDD membutuhkan beberapa perkakas pengembangan sesuai tahapan. Pada tahapan analisis dan perancangan, dibutuhkan perkakas seperti Visual Paradigm yang mampu menggambar model dalam notasi UML. Pada tahap implementasi

(pemrograman) dibutuhkan perkakas pengembangan seperti Play Framework, Yii Framework. Untuk lebih menyempurnakan pengembangan perangkat lunak dibutuhkan perkakas *Integrated Development Environment* (IDE) seperti Eclipse.

Gambar 1 mengilustrasikan pengembangan system komputer menggunakan pendekatan MDD. Pembuat model akan dengan cepat mengetahui apakah model yang dikembangkannya layak diimplementasikan melalui konsep *Forward Engineering*, arah maju dari model menuju kode sumber, dan sebaliknya pekerjaan pembuat kode sumber yaitu Programmer akan dapat diperiksa dengan cepat oleh pembuat model apakah sesuai tidak dengan model yang dikembangkan melalui *reverse engineering*, arah balik dari kode sumber menuju model. Programmer pun bekerja dengan mudah karena sudah ada penuntun pengembangan program berupa nama fungsi/prosedur dalam kesatuan model yang memuat hubungan model beserta spesifikasinya. Pekerjaan programmer adalah mendetilkan dari spesifikasi menjadi kode sumber yang dapat dibaca oleh komputer.

Pengembangan system komputer menggunakan MDD dan dibantu oleh perkakas memberikan kemudahan proses *forward* dan *reverse engineering*. Kebutuhan baru pengembangan system mudah untuk diterapkan, perubahan kode sumber dapat ditelusuri melalui model. Semua langkah dilaksanakan terkomputerisasi dan tidak ada keadaan *magic* untuk menciptakan kode sumber, berbeda halnya dengan pengembangan pendekatan konvensional seperti pengembangan berorientasi fungsi (Grady Booch, 1998).



Gambar 1. Siklus Forward dan Reverse Engineering

Tabel 1. Requirements

No	Dekripsi Requirement	Constraint Candidats
Req.1	Pasien terdiri dari pasien normal dan pasien emergency	
Req.2	Setiap pasien hanya dibolehkan untuk mendapatkan layanan satu kali setiap harinya.	
Req.3	Pasien normal akan mendapatkan layanan kesehatan jika mereka memiliki kartu medis.	
Req.4	Pasien emergency akan mendapatkan layanan kesehatan meskipun tidak memiliki kartu medis.	
Req.5	Setiap pasien memiliki nomor identitas unik untuk mendapatkan layanan kesehatan	
Req.6	Pada satu saat, setiap pasien hanya boleh memiliki satu kartu medis.	
Req.7	Pasien normal yang kartunya hilang tidak dilayani, harus membuat kartu terlebih dahulu.	
Req.8	Pasien, baik yang normal maupun emergency akan dikenakan tagihan setelah selesai dilayani.	
Req.9	Setiap pasien emergency dibolehkan untuk mendapat pelayanan seperti pasien normal	Strong
Req.10	Tak boleh ada satu pun pasien baik yang emergency maupun normal dapat keluar layanan tanpa membayar pembiayaan.	Strong
Req.11	Tak boleh ada satu pasien pun yang telah terdaftar (dapat dikenali sistem) tidak dilayani oleh unit layanan kesehatan	Strong

B. Konstruksi Kebutuhan Prasyarat (Requirement)

Mengacu pada Gambar 1, bagian A merupakan kebutuhan prasyarat (*requirements*). Kebutuhan prasyarat didapat dengan melalukan studi lapangan, survey, wawancara, pengamatan dan perumusan. Pada Tabel 1 terdaftar beberapa kebutuhan prasyarat yang mewakili paket pendaftaran pasien dalam RME. Kebutuhan prasyarat tersebut akan menjadi acuan pengembangan model dan sekaligus sebagai acuan untuk verifikasi model.

C. Pengembangan Metode Formal

M. Famelis (2012) telah menulis paper tentang Model Parsial agar model dapat diverifikasi menggunakan pendekatan formal. Yang dimaksud dengan model dalam paper ini adalah Graph bertipe dengan menggunakan berbagai definisi berikut.

Definisi 1: *Sebuah graph adalah sebuah tuple G=< V; E; s; t >, dimana V adalah himpunan dari node, E adalah himpunan dari edge, dan s, t : E → V adalah fungsi sumber dan target berturut-turut, yang menyatakan setiap edge sebuah node sumber dan node target.*

Definisi 2: *Sebuah grap bertipe (model) dari type T adalah sebuah triple <G; type; T > yang terdiri dari sebuah grap G, sebuah metamodel T, dan fungsi type : G → T yang menyatakan type-type elemen dari G.*

Definisi 3: *Sebuah model parsial adalah sebuah tuple/record* $\langle G, vm, em, \emptyset \rangle$, *dimana* $G = \langle\langle V, E, s, t\rangle, type\rangle$ *adalah type Graph lengkap,* $vm : V \rightarrow \beta$ *dan* $em : E \rightarrow \beta$, *dimana* β *adalah himpunan* $\{True, False, Maybe\}$, *vm dan em adalah*

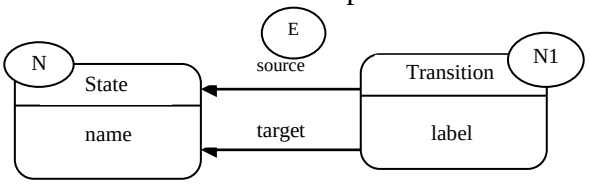
fungsi untuk anotasi atom/elemen dalam G, dan ∅ adalah formula proposisi may dalam ruang lingkup S = V U E.

Definisi 4: *Misalkan M_1, M_2 adalah parsial model, dimana $M_i = \langle G_i, vm_i, em_i, \emptyset_i \rangle$ dengan $G_1 = G_2$. Parsial model M_1 lebih abstrak dari M_2,dinotasikan dengan $M_2 \preceq M_1$ jika dan hanya jika $C(M_2) \subseteq C(M_1)$. $C(M_2)$ dan $C(M_1)$ merupakan model-model konkret yang merepresentasikan model M. Untuk merepresentasikan model, didefinisikan 2 normal form yaitu Graphical Normal Form (GNF) dan Propositional Normal Form (PNF). GNF merepreentasikan informasi dalam bentuk grap, sementara itu PNF mereprsentasikan informasi dalam bentuk formula.*

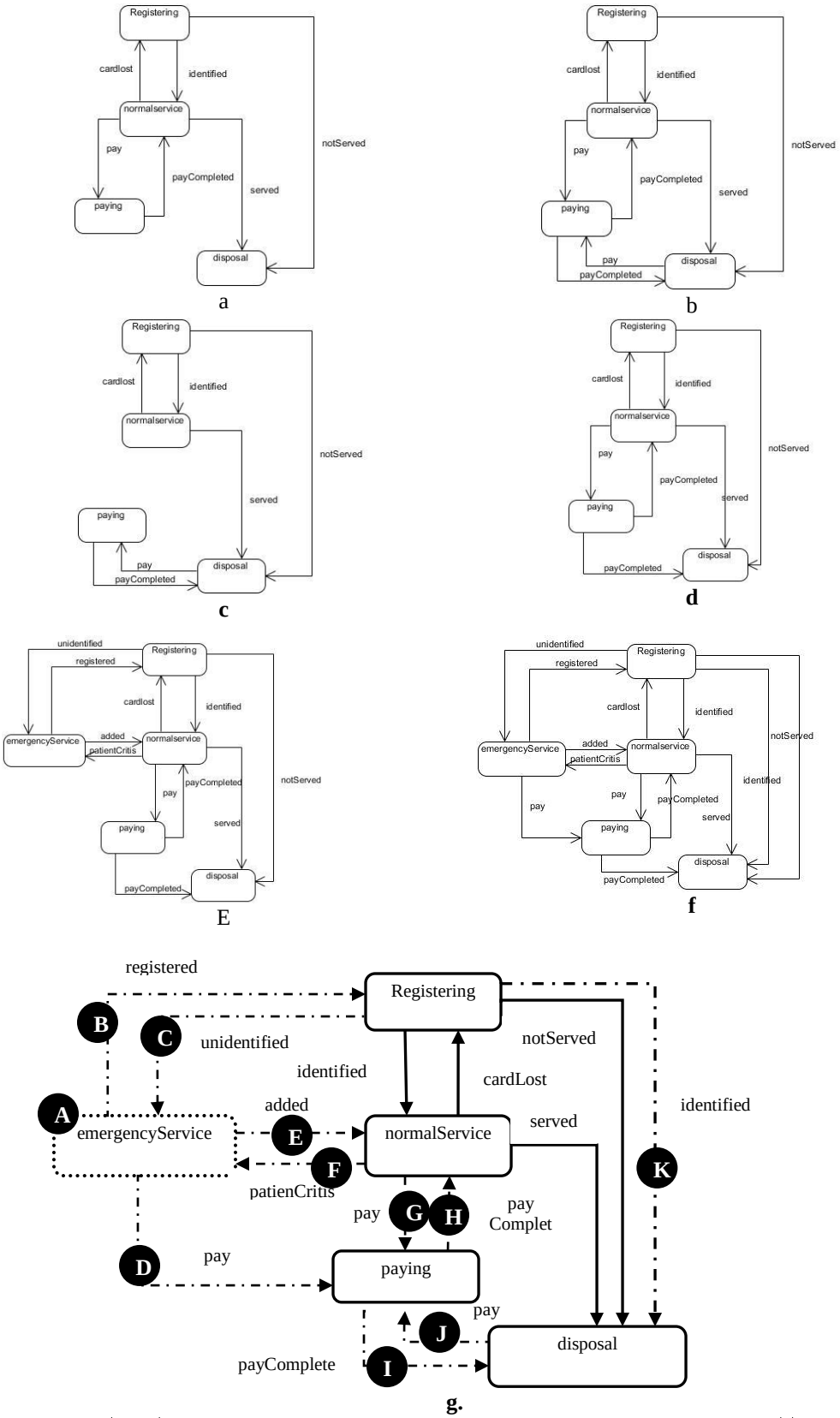
Definisi 5: *Diberikan sebuah parsial model $M = \langle G , \emptyset \rangle$, PNF M adalah sebuah parsial model $M^{PNF} = \langle G^{PNF}, \emptyset^{PNF} \rangle$ yang dibentuk sebagai berikut : 1). $G^{PNF} \subseteq G$ dan ruang lingkup S dari M^{PNF} adalah minimal, 2). Semua elemen di dalam G^{PNF} dinotasikan dengan Maybe,* 3). $\emptyset^{PNF} \models \emptyset$.

Misalkan M merupakan sebuah parsial model dan M^{PNF} merupakan hasil dari definisi 5, maka $M \sim M^{PNF}$.

Langkah untuk memverifikasi model, yaitu langkah pertama membentuk model dalam *State Machine* diagram (SM). Langkah kedua, menterjemahkan SM alternatif rancangan ke dalam perumusan graf. Langkah ketiga, mengembangkan *reasoning*. **Langkah pertama** : Gambar 3 merupakan diagram SM dari enam alternative rancangan perilaku pelayanan pasien dalam RME. SM merupakan aspek dinamis dari struktur class dalam notasi UML. SM terdiri dari *State* dan *Transition*, state digambarkan dengan kotak, *transition* digambarkan dengan garis berarah. Metamodel untuk SM dapat digambarkan pada Gambar 2. **Langkah kedua** : SM dalam Gambar 3 diterjemahkan dalam struktur graf lengkap dan kemudian ditulis dalam formula propositional (∅). Konvensi penulisan SM mengacu pada metamodel dalam Gambar 2, dimana nodes(N) bisa merepresentasikan state, sedangkan edges(E) merepresentaikan label transition. SM dalam Gambar 3 dapat dituliskan dalam variable propositional "N_T" dengan sebuah elemen node N dari Transition T; "E_N1_N2_T" sebagai sebuah elemen E dari sebuah Transition T dengan node asal N1 dan node target N2. Perumusan formula proposisi model dalam Gambar 3 dapat dilihat dalam Tabel 2.



Gambar 2. Metamodel yang digunakan untuk mendefinisikan State Machines



Gambar 3. (a – f) Alternatif Rancangan Perilaku Pelayanan Pasien dalam RME; (g) sebuah Model Parsial M_e dari keenam alternatif tersebut.

Tabel 2. Formula Proposisi dari Model dalam Gambar 3 (a – e)

No	*Propositional Formula*	Keterangan
a $\emptyset_a$ =	Registering_State ∧ normalService_State ∧ paying_State ∧ disposal_State ∧ identified_Registering_normalService_transition ∧ notServed_Registering_disposal_Transition ∧ cardLost_normalService_Registering_Transition ∧ pay_normalService_paying_Transition ∧ payCompleted_paying_normalService_Transition ∧ served_normalService_disposal_Transition ∧ pay_disposal_paying_Transition ∧ payCompleted_paying_disposal_Transition	
b $\emptyset_b$ =	Registering_State ∧ normalService_State ∧ paying_State ∧ disposal_State ∧ identified_Registering_normalService_transition ∧ notServed_Registering_disposal_Transition ∧ cardLost_normalService_Registering_Transition ∧ pay_normalService_paying_Transition ∧ payCompleted_paying_normalService_Transition ∧ served_normalService_disposal_Transition ∧ ¬ pay_disposal_paying_Transition ∧ ¬ payCompleted_paying_disposal_Transition	Gambar 3 (b) dapat dinyatakan dalam ruang lingkup Gambar 3 (a). Terdapat negasi dari pay_disposal_paying_ Transition, dan payCompleted_paying _disposal_Transition
e $\emptyset_e$ =	Registering_State ∧ normalService_State ∧ paying_State ∧ disposal_State ∧ identified_Registering_normalService_transition ∧ notServed_Registering_disposal_Transition ∧ identified_Registering_disposal_Transition ∧ cardLost_normalService_Registering_Transition ∧ pay_normalService_paying_Transition ∧ payCompleted_paying_normalService_Transition ∧ served_normalService_disposal_Transition ∧ payCompleted_paying_disposal_Transition ∧ unidentified_Registering_emergencyService_Transition ∧ registered_emergencyService_Registering_Transition ∧ added_emergencyService_normalService_Transition ∧ patienCritis_normalService_emergencyService_Transition ∧ ¬ pay_emergencyService_paying_Transition	Gambar 3 (e) dapat dinyatakan dalam ruang lingkup Gambar 3 (f).

Langkah ketiga : Mengkonstruksi model parsial dari sekumpulan alternatif model yang telah ditemukan. Untuk membangun model parsial digunakan Algoritma 1 berikut.

Algoritma 1. Mengkonstruksi Model Parsial (M. Famelis dkk, 2012)

Input : Himpunan A merupakan model konkrit $m_i, i \in [0..n-1]$

Output : Sebuah Model Parsial M = $\langle G_M, \Phi_M \rangle$.

1. Konstruksi G_M sebagai gabungan semua $m_i \in$ A
2. Anotasikan elemen yang tidak umum dalam G_M sebagai Maybe.
3. Tetapkan Φ_M = False
4. **for all** $m_i \in$ A, i ∈ [0..n-1] **do**
5. $\emptyset_i$ = True
6. **for all** $e_j \in G_M$, e_j dianotasikan dengan **Maybe**, j ∈ [0..n-1] **do**
7. $\emptyset_i = \emptyset_i \wedge e_j$, dimana jika $e_j \notin m_i$, elemen tersebut harus negasi
8. **end for**
9. $\Phi_M := \Phi_M \; V \; \emptyset_i$
10. **end for**
11. Return M =$\langle G_M, \Phi_M \rangle$

Pembahasan Algoritma : Baris 1 dan 2 pada Algoritma 1 akan menghasilkan model gabungan dari semua model dalam Gambar 3 (a – f). Pada Gambar 3 (g), semua elemen yang tidak umum (*nodes* dan *edges*) ditulis dengan garis terputus-putus, dan menurut **Algoritma 1** dinyatakan sebagai elemen *Maybe*. **Algoritma 1** hanya memproses elemen yang bersifat *Maybe*. Baris 4 sampai dengan 7 akan menghasilkan formula sebagai berikut :

Asumsi : $\emptyset_a = \emptyset_0, \emptyset_b = \emptyset_1, \emptyset_c = \emptyset_2, \emptyset_d = \emptyset_3, \emptyset_e = \emptyset_4, \emptyset_f = \emptyset_5$
e_0 = A, e_1 = B, e_2 = C, e_3 = D, e_4 = E, e_5 = F, e_6 = G, e_7 = H, e_8 = I, e_9 = J, e_{10} = K

$\emptyset_0$=¬ e_0 ∧ ¬ e_1 ∧ ¬ e_2 ∧ ¬ e_3 ∧ ¬ e_4 ∧ ¬ e_5 ∧ e_6 ∧ e_7 ∧ e_8 ∧ e_9 ∧ ¬ e_{10} atau $\emptyset_a$ = ¬ A ∧ ¬ B ∧ ¬ C ∧ ¬ D ∧ ¬ E ∧ ¬ F ∧ G ∧ H ∧ I ∧ J ∧ ¬ K	(2.1)
$\emptyset_1$=¬ e_0 ∧ ¬ e_1 ∧ ¬ e_2 ∧ ¬ e_3 ∧ ¬ e_4 ∧ ¬ e_5 ∧ e_6 ∧ e_7 ∧ ¬ e_8 ∧ ¬ e_9 ∧ ¬ e_{10} atau $\emptyset_b$ = ¬ A ∧ ¬ B ∧ ¬ C ∧ ¬ D ∧ ¬ E ∧ ¬ F ∧ G ∧ H ∧ ¬ I ∧ ¬ J ∧ ¬ K	(2.2)
$\emptyset_2$=¬ e_0 ∧ ¬ e_1 ∧ ¬ e_2 ∧ ¬ e_3 ∧ ¬ e_4 ∧ ¬ e_5 ∧ ¬e_6 ∧ ¬ e_7 ∧ e_8 ∧ e_9 ∧ ¬ e_{10}atau $\emptyset_c$ = ¬ A ∧ ¬ B ∧ ¬ C ∧ ¬ D ∧ ¬E ∧ ¬ F ∧ ¬G ∧ ¬H ∧ I ∧ J ∧ ¬ K	(2.3)
$\emptyset_3$=¬ e_0 ∧ ¬ e_1 ∧ ¬ e_2 ∧ ¬ e_3 ∧ ¬ e_4 ∧ ¬ e_5 ∧ e_6 ∧ e_7 ∧ e_8 ∧ ¬e_9 ∧ ¬ e_{10} atau $\emptyset_d$ = ¬ A ∧ ¬ B ∧ ¬ C ∧ ¬ D ∧ ¬E ∧ ¬ F ∧ G ∧ H ∧ I ∧ ¬ J ∧ ¬ K	(2.4)
$\emptyset_4$= e_0 ∧ e_1 ∧ e_2 ∧ ¬ e_3 ∧ e_4 ∧ e_5 ∧ e_6 ∧ e_7 ∧ e_8 ∧ ¬e_9 ∧ ¬ e_{10} atau $\emptyset_e$ = A ∧ B ∧ C ∧ ¬ D ∧ E ∧ F ∧ G ∧ H ∧ I ∧ ¬ J ∧ ¬ K	(2.5)
$\emptyset_5$= e_0 ∧ e_1 ∧ e_2 ∧ e_3 ∧ e_4 ∧ e_5 ∧ e_6 ∧ e_7 ∧ e_8 ∧ ¬e_9 ∧ e_{10} atau $\emptyset_f$ = A ∧ B ∧ C ∧ D ∧ E ∧ F ∧ G ∧ H ∧ I ∧ ¬ J ∧ K	(2.6)

Menurut baris 6 Algoritma 1, operasi $\emptyset_0 V \emptyset_1 V \emptyset_2 V \emptyset_3 V \emptyset_4 V \emptyset_5 V \emptyset_6$ menghasilkan formula sebagai berikut, dengan terlebih dahulu (2.3) dikonjungsikan dengan (2.4), (2.5) dikonjungsikan dengan (2.6).

((¬ D V D) ∧ (A ∧ B ∧ C ∧ E ∧ F ∧ G ∧ H ∧ I ∧ ¬ J ∧ K)) V
((J V ¬ J) ∧ (¬ A ∧ ¬ B ∧ ¬ C ∧ ¬ D ∧ ¬ E ∧ ¬ F ∧ G ∧ H ∧ I ∧ ¬ K)) V
((I V ¬ I) ∧ (¬ A ∧ ¬ B ∧ ¬ C ∧ ¬ D ∧ ¬E ∧ ¬ F ∧ G ∧ H ∧ ¬ J ∧ ¬ K)) (2.7)

Langkah keempat : Memverifikasi model bertujuan untuk menjawab pertanyaan " apakah property (*requirements*) yang diharapkan dapat dipertahankan ?". Untuk menjawab itu perlu *reasoning*, model parsial M harus dinyatakan dalam Propositional Normal Form (PNF) melalui formula proposisi Φ_M dan property yang diekspresikan sebagai formula proposisi Φ_p.

Kemudian, dilakukan pengecekan satisfiability dari ekspresi $\Phi_M \wedge \Phi_p$ dan $\Phi_M \wedge \neg \Phi_p$, menggunakan dua query **SAT Solver** yang digambarkan dalam Tabel 3. Sebagai contoh, akan diuji salah satu *requirements* penting yang terdapat dalam Tabel 1.

Tabel 3. Pengecekan Property p pada Model Parsial M (A. Biere, 2007).

$\Phi_M \wedge \neg \Phi_p$	$\Phi_M \wedge \Phi_p$	**Property p**
SAT	SAT	Maybe
SAT	UNSAT	True
UNSAT	SAT	False
UNSAT	UNSAT	(*error*)

Langkah awalnya adalah memilih *requirement* yang paling penting, kemudian kalimat *requirement* tersebut diterjemahkan ke dalam formula proposisi. Dipilih req.10 : "Tak boleh ada satu pun pasien baik yang emergency maupun normal dapat keluar layanan tanpa membayar pembiayaan". Melalui pemecahan SAT Solver ditemukan Gambar 3 (a, d, e, f) yang memenuhi $\boldsymbol{\Phi_M} \wedge \neg \boldsymbol{\Phi_p}$, dan terdapat beberapa gambar meliputi Gambar 3**(b, c)** yang memenuhi $\boldsymbol{\Phi_M} \wedge \boldsymbol{\Phi_p}$. Dengan demikian nilai dari req.10 adalah tidak yakin (***Maybe***) pada model.

Dipilih lagi req.11 : "Tak boleh ada satu pasien pun yang telah terdaftar (dapat dikenali sistem) tidak dilayani oleh unit layanan kesehatan". Melalui pemecahan SAT Solver ditemukan Gambar 3 (f) yang memenuhi $\boldsymbol{\Phi_M} \wedge \neg \boldsymbol{\Phi_p}$, dan terdapat beberapa gambar meliputi Gambar 3**(a, b, c, d, e)** yang memenuhi $\boldsymbol{\Phi_M} \wedge \boldsymbol{\Phi_p}$. Dengan demikian nilai dari req.11 adalah tidak pasti (***Maybe***) pada model.

Melalui pemecahan SAT Solver, req.10 dan req.11 dicek kepastiannya, apakah dapat dipertahankan ? Dan jawabannya adalah *Maybe* yang berarti kurang meyakinkan (*uncertain*), oleh karena itu model perlu diperbaiki lagi sampai menghasilkan SAT Solver bernilai *True*.

4. TANTANGAN MASA DEPAN MDD SECARA PRAKTIS

Begitu *massive* pengembangan peragkat lunak untuk memecahkan berbagai permasalahan besar dalam lingkungan korporasi. Namun sayangnya, pendekatan formal untuk menguji atau mengecek ketepatan perancangan masih minim. Tertalu sulit para developer harus memasukkan rumus-rumus matematika rumit ke dalam bidang pekerjaan mereka.

Saat ini beberapa peneliti, seperti N. Ubayashi (2013), telah menyusun pendekatan praktis untuk mengecek tingkat abstraksi sebuah model terhadap program. Walaupun penyusunan model dan translasi model sudah menggunakan perkakas, namun untuk menentukan tingkat abstraksi model (*level of uncertainty*) masih dilakukan secara coba-coba *(try and error*). Penelitian dari N. Ubayashi

(2013) dapat digunakan untuk membangun perkakas MDD berbasis pada pendekatan formal.

Kebiasaan para developer membangun perangkat lunak mengikuti beberapa metode seperti RUP, Agile. Secara umum metode tersebut mengharuskan ada tahapan pengembangan meliputi analisis, *design*, *programming*, *testing*. Pengembangan masing-masing tahapan telah didukung oleh berbagai perkakas modern.

Permasalahannya, perkakas pada tahap analisis dan design belum mampu untuk memeriksa ketepatan design secara atomatis. Kalaupun akan dilakukan pemeriksaan ketepatan design, para developer harus mau rumit melaksanakan pengecekan melalui perumusan matematika. Metode penyusunanya adalah sebagai berikut :

1. Developer menyusun model dalam bentuk grafik diagram kelas (Class Diagram) pada UML tools;
2. Notasi UML ditransformasikan ke dalam OCL dan ditambahi *constraint*, kemudian ditranslasikan ke dalam Alloy language (A. Cunha , 2013; K. Anastasakis, 2007; S.M.A. Shah, 2009);
3. Pemeriksaan model dalam Alloy Language menggunakan SMT/SAT Solver;
4. Memperbaiki model dalam Alloy Language;
5. Transformasi model dalam Alloy Language ke dalam UML lagi (Garis dkk, 2011; S.M.A. Shah, 2009).

Berdasarkan literatur penelitian MDD, pengembangan perkakas MDD secara praktis telah mendekati kenyataan. Penelitian dari M. Famelis (2012) telah menuntun para peneliti untuk mengembangkan MDD dengan pendekatan formal, dan penelitian N. Ubayashi (2013) telah membukakan jalan penyusunan *compiler* pengembangan model menggunakan perkakas. Dari kedua penelitian tersebut diharapkan dapat dikembangkan penelitian pengembangan MDD dengan pendekatan formal secara praktis.

5. KESIMPULAN DAN SARAN

Model Rekam Medis Elektronis (RME) yang dikembangkan, dapat diverifikasi secara formal menggunakan pendekatan graf lengkap. Model yang merepresentasikan kebutuhan prasyarat diperbaiki setiap ada hasil pengecekan, antara model yang didisjungsikan dengan *requirements* menggunakan SAT

Solver, berturut-turut dinyatakan dalam Propositional Normal Form $\boldsymbol{\Phi_M}$ dan $\boldsymbol{\Phi_p}$. Jika hasil disjungsi menghasilkan logika ***Maybe*** berarti *requirement* tersebut dapat dipertahankan, tetapi model harus diperbaiki. Jika menghasilkan logika ***True*** berarti *requirement* tersebut sudah pasti (*certain*).

Sebagai saran, karena penelitian ini masih bersifat awal untuk menginvestigasi kepastian model, perlu dilakukan penelitian lanjutan untuk mendiagnosis dan perbaikan (*refinement*) model menggunakan pendekatan formal.

DAFTAR PUSTAKA

Cunha, A. Garis, D. Riesco. (2013) "Translating between Alloy Specifications and UML Class Diagrams Annotated with OCL", Springer.

Biere. (2007) "Short History on SAT Solver Technology and What is Next?, invited talk," in Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'07), 2007, slides available at http://fmv.jku.at/biere/talks/Biere-SAT07-talk.pdf, diakses 15 November 2014.

Grady Booch, Ivar Jacobson, and James Rumbaugh. (1998) Unified Modeling Language 1.3, White paper, Rational Rational Software Corp.

Garis, A. Cunha, and D. Riesco. (2011) "Translating Alloy Specifications to UML Class Diagrams Annotated with OCL", Proceeding SEFM'11 Proceedings of the 9th international conference on Software engineering and formal methods, Pages 221-236, Springer-Verlag Berlin, Heidelberg.

K. Anastasakis, B. Bordbar, G. Georg, and I. Ray. (2007) "UML2Alloy: A Challenging Model Transformation," in ACM/IEEE 10th International Conference on Model Driven Engineering Languages and Systems, Nashville, USA, pp. 436-450

M. Famelis, R. Salay, M. Checkik. (2012) "Partial Models: Toward Modeling and Reasoning with Uncertainty", Proceeding ICSE '12, Proceedings of the 34th International Conference on Software Engineering, USA, Pages 573-583.

N. Ubayashi, A. Di, S. Hosoai, Y. Kamei. (2013) "Abstraction-aware Compiler for Yet Another MDD", Lab POSL, Kyushu University, Japan, unpublised.

S.M.A. Shah, K. Anastasakis, and B. Bordbar. (2009) "From UML to Alloy and Back Again", MoDevVa 2009 : Proceedings of the 6th International Workshop on Model-Driven Engineering, Verification and Validation, pp- 1-10. ACM, New York.

Taryana A, Setyono J, Wijayanto B. (2013). Laporan Akhir Stranas Dikti 2013, Penelitian Stranas Dikti 2013, Indonesia.