 ISSN: 1411-3201

PEMODELAN KONTEKS DAN CONTEXT AWARENESS AVATAR DI DUNIA VIRTUAL

Muhammad Dedi Iskandar[1)], Lukito Edi Nugroho[2)], Paulus Insap Santosa[2)]

[1,2,3)] Teknik Elektro dan Teknologi Informasi FT UGM
email : dedi_i@mail.ugm.ac.id

Abstraksi

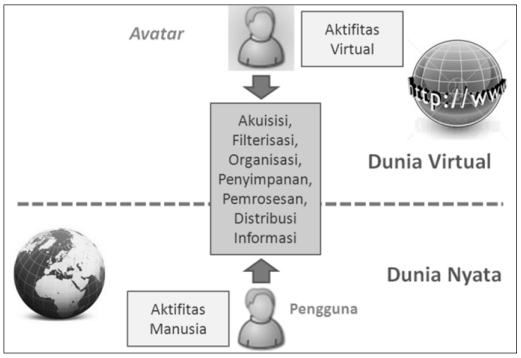
Konteks avatar perlu dimodelkan secara utuh sehingga dapat digunakan dalam beragam skenario dunia virtual yang dinamis. Tool pemodelan yang digunakan harus dapat meminimalisir ambiguitas penafsiran pada pengembangan sistem selanjutnya, sehingga digunakanlah Mocha, suatu tool spesifikasi berbasis Ambient Calculus dan UNITY. Mocha dikembangkan sehingga dapat mengekspresikan situasi dimana dua atau lebih konteks terjadi secara konkuren. Situasi konteks konkuren memiliki bentuk situasi konteks yang unik pada dunia virtual dan tidak dapat dimodelkan oleh notasi konteks primitif.

Kata Kunci :
avatar, pemodelan, konteks, context awareness, dunia virtual.

Pendahuluan

Penggunaan sistem informasi dan komunikasi dalam mendukung kegiatan akademis semakin meningkat baik dari sisi jumlah maupun bentuk layanan. Proses pembelajaran tidak terikat oleh tempat dan waktu dengan adanya teknologi bergerak (menggunakaan perangkat *mobile)* dan lingkungan yang menyediakan jaringan akses layanan dimana-mana (*ubiquitous*). Layanan teknologi informasi yang semakin saling terhubung satu sama lain menciptakan suatu pandangan teknologi informasi sebagai suatu infrastruktur yang diistilahkan dengan dunia virtual (maya). Dengan harapan akan adanya peningkatan produktivitas dengan kepraktisan dan kecepatan yang ditawarkan oleh dunia virtual, maka semakin bermunculan pula layanan yang mentransformasi aktivitas di dunia fisik ke dunia virtual.

Di sisi lain, beban kerja manusia dalam mengelola aktivitas dunia virtualnya sambil sementara melakukan aktivitas dunia fisik ternyata menjadi kontraproduktif karena tidak sebanding dengan kemampuan manusia dalam mengakses, mengelola, dan menyelaraskan aktivitas yang ada di dalam kedua dunia tersebut. Sebagai salah satu solusi terhadap masalah ini, diciptakanlah konsep avatar, yaitu suatu entitas komputasi yang merepresentasikan pengguna manusia di dunia virtual. Sementara avatar bekerja, maka penggunanya dapat melakukan aktivitas produktif lainnya di dunia nyata (Gambar 1).



Gambar 1. Pengguna di Dunia Nyata dan Avatar representasinya di Dunia Virtual

Pada implementasinya, suatu avatar bekerja dan berinteraksi dengan avatar yang lain menggunakan mekanisme *context awareness*. Dalam mekanisme tersebut, konteks berperan sebagai pemicu seluruh aksi yang dapat dilakukan oleh avatar. Penelitian ini mengembangkan model konteks avatar. Pemodelan konteks merupakan hal yang penting dalam implementasi sistem avatar karena keunikan dari sistem avatar terhadap sistem *context aware* pada umumnya. Model konteks harus direpresentasikan secara generik agar dapat bekerja pada dunia virtual yang memiliki jenis skenario interaksi yang beragam dan dinamis. Tool pemodelan yang digunakan juga harus dapat menghasilkan model yang jelas dan tepat untuk meminimalisir terjadinya ambiguitas penafsiran pada proses pengembangan sistem selanjutnya. Beberapa situasi konteks tertentu terkait *context awareness*, seperti terjadinya dua konteks atau lebih pada avatar pada suatu waktu juga perlu dimodelkan.

 ISSN: 1411-3201

Tinjauan Pustaka

Pengembangan kerangka kerja pemodelan pada penelitian [4] mendukung perancangan model *context-awareness* pada komputasi bergerak (mobile computing). Generisitas dan formalitas dari kerangka kerja bernama *Mocha* tersebut membuatnya dapat diperluas lebih lanjut untuk memodelkan mekanisme interaksi sistem *context aware* pada umumnya. Penelitian ini mencoba menggunakan dasar dari *Mocha* untuk memodelkan avatar dalam interaksinya. Sementara *Mocha* hanya mendukung ekspresi *context awareness* untuk konteks individual, maka penelitian ini mengembangkan *Mocha* untuk dapat mendukung ekspresi konteks yang lebih kompleks, yaitu situasi terjadinya lebih dari satu konteks pada avatar pada waktu yang bersamaan.

Pemodelan interaksi pada [5] merupakan ide dasar dari pemodelan konteks berbasis aktivitas dan role avatar. Interaksi yang dimodelkan dalam penelitian ini, dikembangkan berbasis peran avatar dalam suatu konteks. Namun demikian, model yang dijelaskan penelitian [5] tersebut telah spesifik diarahkan untuk suatu model arsitektur sistem tertentu. Pemodelan interaksi dalam penelitian ini membahas avatar dalam tingkat konseptual, sehingga dapat diterapkan dalam beragam arsitektur dan tool yang lebih spesifik pada kegiatan pengembangan implementatif.

Sistem Context-Aware

Dalam interaksi antar manusia, sejumlah besar informasi dapat tersampai tanpa dibutuhkan komunikasi secara eksplisit. Gesture, ekspresi wajah, hubungan dengan orang atau obyek lain disekitarnya dan pengetahuan bersama tentang masa lalu semua digunakan sebagai petunjuk yang membantu komunikasi [2]. Informasi implisit yang disebut konteks tersebut, membuat bandwidth percakapan menjadi lebih efisien.

Sementara dalam interaksi manusia dengan komputer, konteks yang dapat digunakan bersama sangatlah minim. Konteks dalam diri alamiah manusia, seperti bentuk sinyalemen kebahagiaan atau kebencian, lebih sulit untuk diproses oleh komputer. Sementara contoh konteks yang telah umum digunakan oleh komputer misalnya, lokasi pengguna (melalui sistem lokalisasi) yang dapat digunakan untuk pencarian posisi. Teknologi sensor menjadi sumber utama informasi konteks dari lingkungan, sehingga konteks yang dapat diterima oleh sistem tergantung pada batasan teknologi sensor yang ada.

Konsep Avatar

Menurut [1], dalam sistem komputer interaktif, karakter secara tradisional diwujudkan dalam dua bentuk : avatar dan agent. Suatu avatar adalah sebuah karakter yang merepresentasikan dan dikontrol oleh manusia. Sebagai contoh, dalam sebuah video game, karakter yang dikendalikan oleh pemain adalah avatar. Kebalikannya, agent adalah suatu karakter yang dikendalikan oleh komputer dengan suatu algoritma tertentu []. Semua karakter 'alien' yang harus dibasmi dalam permainan video game 'Alien 3' dari mesin console Super Nintendo Entertainment System tahun 1990 merupakan contoh dari agent. Alien-alien tersebut mengawasi aksi tokoh utama dan bertindak untuk melawan dengan dikendalikan oleh program komputer.

[3] menjelaskan bahwa avatar pada implementasinya digunakan sebagai karakter representasi pengguna untuk beraktifitas dan berkomunikasi dengan avatar yang lain pada dunia virtual. Namun adakalanya, suatu karakter merupakan kombinasi dari avatar dan agent. Pengguna manusia menginisiasi perintah pada avatarnya lalu kemudian avatar tersebut akan melakukan aktifitasnya sesuai algoritma tertentu dengan parameter yang ditentukan oleh penggunanya. Pada penelitian ini, entitas avatar sebagai model representasi dari penggunanya juga memiliki 'kecerdasan' dalam memilih dan menjalankan suatu aksi seperti halnya sebuah agent cerdas. Dengan kata lain, avatar yang digunakan adalah jenis hybrid. Ada kalanya avatar terikat atau mencerminkan situasi penggunanya (bounded), namun di saat lain avatar dapat melakukan aktifitas virtual pengguna sementara penggunanya melakukan hal yang lain (unbounded).

Mobile Ambient Calculus

Mobile Ambient adalah sebuah kalkulus untuk komputasi bergerak yang berdasarkan pada konsep suatu ambien atau lingkungan terbatas [4]. Walaupun menggunakan istilah 'lingkungan', yang dimaksud dengan ambien pada umumnya merujuk pada suatu entitas yang dibatasi oleh struktur fisiknya. Seperti contohnya suatu entitas halaman web, dibatasi oleh struktur filenya atau suatu space alamat Internet, dibatasi oleh suatu range alamat IP. Suatu ambien memiliki karakteristik struktural, yaitu suatu ambient dapat mengandung satu atau lebih ambient lainnya sehingga dapat membentuk suatu hirarki.

Sintaksis dari kalkulus tersebut secara lengkap didefinisikan sebagai berikut ini. Sintaksis tersebut melibatkan dua proses (P dan Q) dan kapabilitas (M) sebagai komponen utamanya.

P, Q	::=	0	inaktivitas
	|	$(\nu n)P$	restriksi
	|	$P \mid Q$	komposisi
	|	$!P$	replikasi
	|	$n[P]$	ambien
	|	$M.P$	aksi
	|	n	nama

 ISSN: 1411-3201

$$
\begin{array}{lll}
M & ::= & in\ n \quad \text{dapat masuk ke } n \\
 & | & out\ n \quad \text{dapat keluar dari } n \\
 & | & open\ n \quad \text{dapat membuka } n
\end{array}
$$

Ambien dalam sintaks tersebut dapat memiliki struktur $n[P]$, dimana *n* adalah nama dari suatu ambien dengan proses *P* berada di dalamnya. *P* sendiri adalah suatu proses yang independen terhadap proses atau operasi lain (misalnya terhadap *Q*) sehingga *P* dapat secara aktif berjalan secara paralel bersamaan dengan proses atau operasi lainnya, termasuk yang yang berada pada ambien hostnya. Suatu *kapabilitas* adalah aksi atau operasi yang dapat dilakukan oleh suatu ambien. Notasi *M.P* merepresentasikan suatu aksi yang didefinisikan oleh kapabilitas *M*, dan berjalan terhadap proses *P*. Ada tiga jenis kapabilitas, yaitu untuk masuk (*in*) ke suatu ambien, keluar (*out*) dari ambien, dan buka (*open*) suatu ambien.

Mekanisme dari aksi dapat dicontohkan sebagai berikut, $in\ m.P$, memiliki arti bahwa kapabilitas *in.m* mengatakan pada ambien yang melingkupi aksi $in\ m.P$ untuk memasuki ambien *m*. *Behaviour* dari kapabilitas ini ditunjukkan oleh suatu reduksi sebagai berikut :

$$n[in\ m.P|Q] \mid m[R] \rightarrow m[n[P|Q]|R]$$

Dalam reduksi tersebut dapat diterangkan bahwa ambien *n*, yang mengandung proses *P* dan *Q*, adalah sibling dari ambien yang lain yaitu *m* yang mengandung proses *R*. Aksi $in\ m.P$, jika berhasil dieksekusi, akan mengubah *n* sebagai anak dari ambien *m*, diikuti dengan P. Q sebagai proses lain di dalam *n* juga akan mengikuti.

Suatu kapabilitas keluar *out.m* dalam suatu aksi *out m.P* akan mengakibatkan ambien yang melingkupi aksi tersebut keluar dari ambien parent yang bernama *m*. Reduksi berikut menunjukkan proses tersebut :

$$m[n[out\ m.P|Q]|R] \rightarrow n[P|Q]|m[R]$$

Kapabilitas keluar merupakan kebalikan dari kapabilitas masuk, pada reduksi di atas, jika operasi *out.m* tersebut berhasil, maka akan mengeluarkan ambien *n* dari parent nya, yaitu ambien *m*, diikuti proses P itu sendiri dan siblingnya yaitu Q menuju hirarki ambien yang lebih tinggi.

Kapabilitas terakhir adalah membuka suatu ambient, yaitu *open n*. Jika digunakan dalam sebuah aksi *open n.P*, maka ia akan menghilangkan ambien bernama *n* yang berada pada level yang sama dengan aksi tersebut. Jika *n* memiliki anak, maka anak tersebut akan lepas dari batasan *n*. Dalam reduksi di bawah ini, digambarkan bagaimana aksi *open* tersebut bekerja.

$$open\ n.P|n[Q] \rightarrow P|Q$$

UNITY

UNITY adalah suatu model komputasional dan suatu sistem pembuktian untuk spesifikasi program. Pemodelan sistem dengan UNITY menggunakan suatu formalisme berbasis logika temporal linear. Formalisme tersebut berdasarkan suatu pernyataan dalam bentuk $\{p\}s\{q\}$. Suatu pernyataan dengan format demikian memiliki arti bahwa suatu eksekusi dari suatu aksi *s* dalam state apapun yang memenuhi predikat *p* akan menghasilkan suatu state yang memenuhi predikat *q*. Format ini digunakan untuk menerangkan properti dari suatu program, yang dibagi menjadi properti *safety* dan *progress*. Properti *safety* menentukan state-state yang diperbolehkan untuk dimiliki oleh suatu program, sementara properti *progress* menerangkan semantik operasional dari transisi suatu state. Spesifikasi formal dari suatu dari suatu program UNITY dapat dinyatakan menggunakan properti *safety* dan *progress* dari program, yang ditulis dalam bentuk kumpulan pernyataan-pernyataan logika.

Relasi	Arti
p **unless** q	Jika p terpenuhi ketika q tidak terpenuhi, maka ia akan terus demikian selagi q tidak terpenuhi
stable p	Didefinisikan sebagai p **unless** *false*
inv p	Didefinisikan sebagai $(\textbf{INIT} \rightarrow p) \wedge (\textbf{stable}\ p)$
$p \mapsto q$	Jika p terpenuhi, maka kemudian q akan terpenuhi. Ketika q sudah terpenuhi maka ia tidak bergantung pada p apakah tetap terpenuhi atau tidak.
p **until** q	Didefinisikan sebagai $(p \mapsto q) \wedge (p\ \textbf{unless}\ q)$
p **ensures** q	Relasi ini akan terpenuhi jika p **unless** q terpenuhi; dan ada suatu aksi yang apabila dieksekusi ketika $(p \wedge \sim q)$ bernilai true, akan membuat q bernilai true; dan aksi tersebut pasti terjadi kemudian.

Tabel 1. Relasi logika UNITY

UNITY memperkenalkan beberapa relasi predikat dasar sebagai bagian dari sistem logikanya, yaitu : **unless**, **ensures**, **leads-to** (disimbolkan dengan " ↦ "). Ketiga relasi tersebut kemudian dapat diturunkan menjadi relasi lain. Relasi-relasi tersebut diterangkan dalam tabel 1.

UNITY juga memperkenalkan suatu pernyataan konstruktor yang terdiri dari tiga bagian yang dapat dituliskan dalam sintaks berikut ini :

 ISSN: 1411-3201

$$\langle \textbf{op}\ variables : range\_constraint :: statement\_expr \rangle \quad (2.1)$$

Operator **op** dapat berupa suatu operator biner, asosiatif, atau komutatif, seperti Σ (penjumlahan kumulatif), Π (perkalian kumulatif), ∀ (kuantifier universal), dan ∃ (kuantifier eksistensial). Konstruktor menghasilkan suatu multi-set nilai $(v_1, v_2, \dots, v_n)$ dengan secara logika mengevaluasi *statement expression* untuk setiap variabel di dalam *range constraint*. Nilai akhir yang dihasilkan oleh konstruktor didapat dengan mengevaluasi keseluruhan ekspresi berdasarkan operator yang digunakan.

[6] mengembangkan UNITY dengan menambahkan konsep tempat, waktu dan aksi ke dalam UNITY. Ketiga konsep tersebut dibutuhkan di dalam pemodelan sistem bergerak (*mobile system*). Konsep tempat (dilambangkan dengan 'λ') diperkenalkan dengan mengikutsertakan lokasi sebagai suatu atribut tambahan dalam representasi state. Sementara waktu membawa properti temporal yang dapat digunakan dalam pemodelan *behaviour* dari program yang terkait waktu. Notasi aksi dikembangkan sehingga dapat digunakan untuk elemen *mobile* secara individual. Suatu aksi terikat pada lokasi dan dienkapsulasi dengan relasi **takes-to**. Ekspresi di bawah ini memiliki arti bahwa jika *A* dieksekusi pada λ maka akan mengubah *P* menjadi *Q* ketika *R* terpenuhi.

$$A\ \textbf{at}\ \lambda\ \textbf{takes}\ P\ \textbf{to}\ Q\ \textbf{when}\ R \quad (2.2)$$

Metode Penelitian

Penelitian ini dilakukan mengikuti alur sebagai berikut:

1. Identifikasi masalah dan studi literatur.
   Untuk mendapatkan pemahaman yang mendalam terhadap permasalahan yang dibahas pada penelitian ini, maka tahap awal dalam mengidentifikasi masalah dilakukan dengan studi literatur terhadap penelitian-penelitian tentang sistem pervasive dan context-awareness terutama yang terkait dengan pemodelan interaksinya. Selain untuk menemukan permasalahan, dalam tahap ini peneliti juga mencoba menemukan beberapa tool pemodelan yang juga telah dibahas oleh peneliti-peneliti sebelumnya untuk kemudian dikembangkan sesuai kebutuhan penelitian ini.
2. Pengembangan model representasi konteks.
   Konteks dimodelkan sejauh mungkin meminimalisir timbulnya ambiguitas dalam menginterpretasi model dan memudahkan pengembangan sistem aplikasi, maka digunakan tool pemodelan yang berbasis bahasa formal. Pemodelan konteks menggunakan dan mengembangkan tool pemodelan Mocha [4], yaitu suatu bahasa pemodelan berbasis Ambient Calculus dan UNITY untuk sistem komputasi bergerak berbasis logika temporal. Ambient Calculus merupakan kalkulus untuk komputasi mobile yang didasarkan pada konsep ambien atau entitas terbatas. Ambient calculus memberikan kontribusi dalam mendefinisikan struktur hirarkis dari entitas avatar. UNITY merupakan suatu sistem pembuktian untuk suatu spesifikasi program yang dapat digunakan dalam membuat model komputasional. UNITY menggunakan logika temporal linear dalam memberikan spesifikasi program yang terdiri dari properti safety dan progress.
3. Pengembangan tool representasi konteks
   Pengembangan tool dilakukan sehingga Mocha tidak hanya bekerja pada konteks individual, namun juga dapat mengekspresikan situasi konteks yang lebih kompleks, yaitu situasi dimana dua atau lebih konteks terjadi secara bersama-sama atau konkuren. Situasi konteks konkuren memiliki bentuk situasi yang unik dan tidak dapat dimodelkan oleh notasi konteks primitif. Skenario interaksi yang membutuhkan ekspresi situasi konteks konkuren tersebut juga disampaikan pada publikasi ini.
4. Pembahasan hasil penerapan pemodelan interaksi berbasis konteks.
5. Pengambilan kesimpulan dan penyusunan laporan.

Hasil dan Pembahasan

Model Konteks Dasar

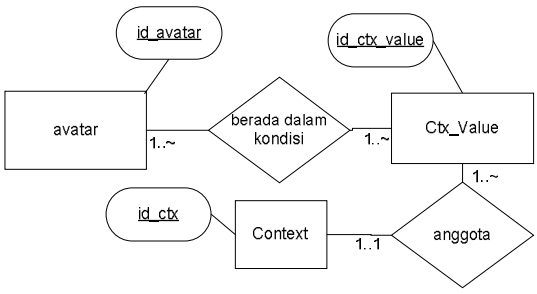
Himpunan seluruh domain konteks $\mathbb{C}$ pada sistem virtual dirumuskan sebagai berikut:

$$\mathbb{C} = \{C_1, C_2, \dots, C_j, \dots\}$$

dimana $C_j, 1 \leq j \leq \infty$ adalah sebarang domain konteks tunggal. Suatu domain konteks C_j didefinisikan sebagai himpunan nilainya. Nilai konteks merepresentasikan seluruh kondisi yang ada dalam suatu konteks sebagai berikut:

$$C_j = \{x_{j1}, x_{j2}, \dots, x_{ji}, \dots, x_{jm}\}$$

dimana $x_{ji}, 1 \leq i \leq \infty$ adalah sebarang nilai konteks dalam konteks $\boldsymbol{C_j}$.



Gambar 2. Hubungan avatar, nilai konteks dan konteks

 ISSN: 1411-3201

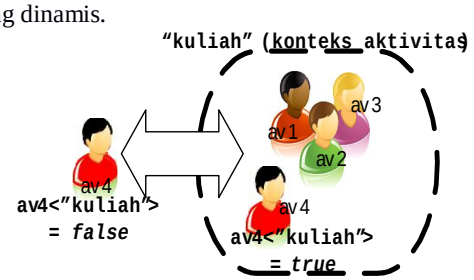
Suatu avatar dapat mengalami situasi yang direpresentasikan oleh suatu nilai konteks. Setiap avatar dapat mengalami banyak nilai konteks dan sebaliknya setiap nilai konteks juga dapat dialami dari beberapa avatar seperti hubungan *many-to-many dalam* E-R Diagram (Gambar 2).

State konteks adalah suatu konsep yang menunjukkan apakah suatu avatar sedang mengalami nilai konteks tertentu atau tidak. State konteks mengaitkan konteks dengan avatar sebagai suatu atribut. Sebagai contoh "berada di dalam kuliah" merupakan contoh salah satu nilai dari domain konteks *aktivitas* di dalam lingkungan kampus. Konteks "berada di dalam kuliah" menjadi state konteks ketika dilekatkan pada suatu avatar sebagai atribut dan menunjukkan apakah avatar tersebut sedang berada dalam kuliah atau tidak.

State konteks hanya memiliki 2 (dua) nilai yang saling berlawanan, yaitu mengalami dan tidak mengalami suatu konteks. Oleh karenanya, state konteks merupakan suatu atribut yang bertipe *boolean*. Jika x_{ji} adalah suatu nilai konteks, maka s_{ji} merupakan state konteks yang berkorelasi dengan x_{ji}. State konteks s_{ji} dimodelkan sebagai atribut pada avatar av sebagai berikut.

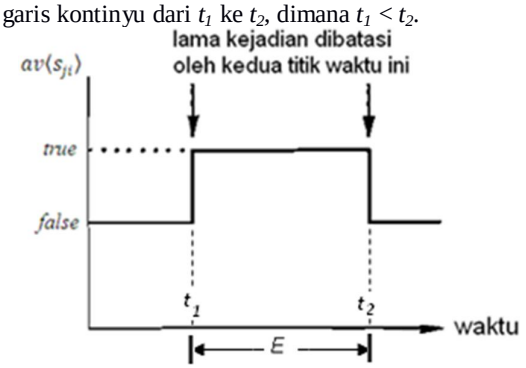
$$av\langle s_{ji}\rangle \equiv av\langle s(x_{ji})\rangle = \begin{cases} true & jika\ av \bullet x_{ji} \\ false & selainnya \end{cases}$$

dimana operator "• " memiliki arti "sedang mengalami" (av sedang mengalami kondisi x_{ji}). Sebagai contoh pada Gambar 3 ditunjukkan posisi avatar $av4$ terhadap aktivitas "kuliah". Ketika avatar *av4* dalam aktivitas kuliah, maka statemen '$av4\langle"kuliah"\rangle$' bernilai *true* dan sebaliknya bernilai *false* ketika avatar *av4* tidak sedang mengalami aktivitas kuliah. Adanya perubahan-perubahan nilai konteks state ini menunjukkan bahwa lingkungan fisik avatar merupakan sesuatu yang dinamis.



Gambar 3. Nilai state konteks berdasarkan posisi avatar terhadap konteks

Dinamika konteks dapat dikaitkan dengan dimensi waktu sehingga suatu state konteks dapat pula digambarkan dalam suatu fungsi waktu. Saat dimana avatar mengalami suatu konteks disebut dengan kejadian konteks. Kejadian konteks digambarkan dalam garis waktu (Gambar 4) sebagai suatu interval *E* dari titik t_1 sampai t_2. Titik t_1 mencerminkan saat terjadi perubahan state konteks $av\langle s_{ji}\rangle$ dari *false* menjadi *true*, sebaliknya titik t_2 adalah saat terjadi perubahan state konteks $av\langle s_{ji}\rangle$ dari *true* menjadi *false*. Durasi kejadian konteks merupakan suatu garis kontinyu dari t_1 ke t_2, dimana $t_1 < t_2$.



Gambar 4. titik perubahan state konteks pada t_1 dan t_2.

Dari penjelasan tersebut dapat disimpulkan bahwa *E* merupakan saat dimana state konteks $av\langle s(x_{ji})\rangle$ bernilai *true* atau dengan kata lain, *E* merupakan saat kejadian ketika av mengalami x_{ji}. Oleh karenanya nilai kebenaran, dari kejadian konteks *E* pada avatar setara dengan nilai kebenaran dari state konteks avatar, yaitu :

$$av\langle E\rangle \equiv av\langle s(x_{ji})\rangle$$

Terhadap suatu kejadian *E*, avatar memiliki tiga state ditinjau dari perspektif waktu (dinamakan *state mobilitas*), yaitu sebelum (*before*) *E*, ketika (*at*) E, dan setelah (*after*) E. Setiap state konteks mobilitas *E* pada avatar didefinisikan dalam berikut:

$$av\langle before(E)\rangle = \begin{cases} true & jika\ (t < t_1) \\ false & selainnya \end{cases}$$

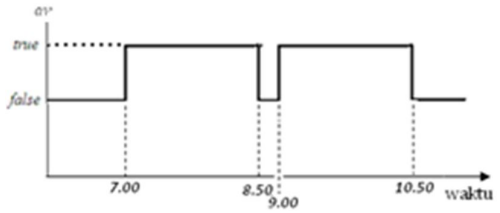
$$av\langle at(E)\rangle = \begin{cases} true & jika\ (t_1 \le t \le t_2) \\ false & selainnya \end{cases}$$

$$av\langle after(E)\rangle = \begin{cases} true & jika\ (t > t_2) \\ false & selainnya \end{cases}$$

Nilai *t* merupakan posisi avatar *av* dalam dimensi waktu. Sesuai dengan sifat alamiahnya, nilai dari *t* akan terus bertambah dan dalam satu waktu avatar pun hanya dapat berada pada satu state kejadian.

Suatu kejadian konteks terjadi tanpa terputus dalam suatu rentang waktu dimulai dari mulai dialaminya kejadian (nilai state konteks berubah dari *false* menjadi *true*) sampai kejadian tersebut berakhir (nilai state konteks berubah dari *true* menjadi *false*) dalam rentang t_1 ke t_2. Oleh karenanya, suatu konteks dianggap telah dialami dan berakhir ketika avatar telah mengalami satu siklus nilai state konteks (*false-true-false*). Ketika siklus tersebut terulang kembali pada state konteks tersebut, maka dapat dikatakan bahwa avatar telah mengalami konteks tersebut lebih dari sekali (Gambar 5).

 ISSN: 1411-3201



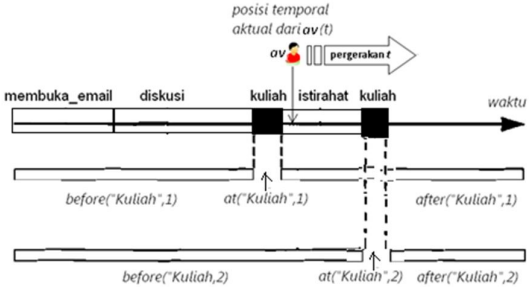
Gambar 5. Avatar *av* mengikuti aktivitas kuliah sebanyak dua kali

Dalam beberapa kasus suatu konteks yang sama dapat terjadi lebih dari sekali pada avatar. Misalnya suatu konteks aktivitas "kuliah" yang diikuti av dalam Gambar 5, pernyataan konteks state $av\langle"kuliah"\rangle = true$ belum dapat mengidentifikasi apakah avatar *av* sedang mengikuti kuliah yang dimulai pada pukul 7.00 ataukah yang dimulai pukul 9.00. Oleh karenanya model konteks dengan notasi x_{ji} di atas dikembangkan menjadi berbentuk (x_{ji}, n_j). Indeks n_j adalah suatu indeks pencacah (*counter*) yang memberi *identitifier* pada konteks tersebut.

Dengan mempertimbangkan: keterkaitan state konteks terhadap dimensi temporal, kejadian *E* merupakan representasi terjadinya suatu nilai konteks pada avatar dan pemberian *identifier* pada kejadian; maka model state konteks avatar didefinisikan kembali sebagai berikut.

$$av\langle before(x_{ji}, n_j)\rangle \triangleq$$
$$av\langle \diamond (x_{ji}, n_j)\rangle = \begin{cases} true & jika\ (t < t_1) \\ false & selainnya \end{cases}$$
$$av\langle at(x_{ji}, n_j)\rangle \triangleq$$
$$av\langle (x_{ji}, n_j)\rangle = \begin{cases} true & jika\ (t_1 \le t \le t_2) \\ false & selainnya \end{cases}$$
$$av\langle after(x_{ji}, n_j)\rangle \triangleq$$
$$av\langle (x_{ji}, n_j) \diamond\rangle = \begin{cases} true & jika\ (t > t_2) \\ false & selainnya \end{cases}$$

dengan simbol $\triangleq$ memiliki makna "disingkat menjadi". Tanda "$\diamond$" menunjukkan state *sebelum* (jika diletakkan di depan notasi state konteks) dan *sesudah* (jika diletakkan di belakang).

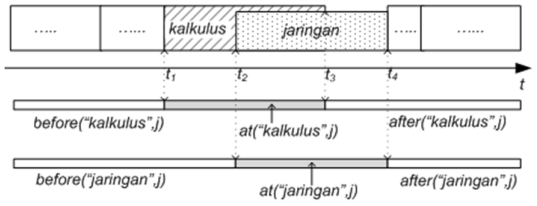


Gambar 6. Dua aktivitas kuliah dengan *identifier* dan posisi temporal avatar

Sebagai contoh, dalam Gambar 6 avatar *av* pada *t* memiliki nilai *true* untuk ekspresi *av<after("Kuliah",1)>* dan *av<before("Kuliah",2)>* yang menandakan avatar *av* berada posisi setelah mengikuti kuliah yang pertama dan sebelum mengikuti kuliah yang kedua.

Konteks Konkuren

Ada kalanya, suatu avatar mengalami suatu situasi dimana lebih dari satu konteks terjadi secara konkuren, misalnya seorang dosen yang melanjutkan studi sebagai mahasiswa pascasarjana berada dalam "kuliah kalkulus" dan beberapa saat kemudian sambil mengikuti perkuliahan ia juga mengajar kuliah jaringan pada mahasiswanya secara online lewat laptop sehingga juga berada dalam "kuliah jaringan" . Konkurensi state konteks terjadi pada saat suatu avatar berada dalam "kuliah_kalkulus" dan "kuliah_jaringan" dalam waktu bersamaan.



Gambar 7 Dua konteks state yang mempunyai konkurensi

Model konteks konkuren yang dipresentasikan dalam penelitian ini mengembangkan notasi konteks primitif yang ada pada *Mocha*. Penelitian ini memanfaatkan elemen temporal (waktu) untuk memodelkan konkurensi konteks. Diberikan T_i dan T_j adalah dua himpunan beranggotakan setiap elemen waktu dimana secara berturut-turut state konteks $av\langle s_i\rangle$ dan $av\langle s_j\rangle$ bernilai *true*. Pernyataan tersebut dapat dituliskan sebagai berikut : **(4.6)**

$$T_i = \{t_1^i, t_2^i, .., t_m^i | av\langle s_i\rangle = true\} \text{ dan}$$
$$T_j = \{t_1^j, t_2^j, .., t_n^j | av\langle s_j\rangle = true\}$$

Dari definisi tersebut, maka adanya konkurensi antara $av\langle s_i\rangle$ dan $av\langle s_j\rangle$ dapat dinotasikan dengan $HasConc(av\langle s_i\rangle, av\langle s_j\rangle)$, dimodelkan sebagai berikut:

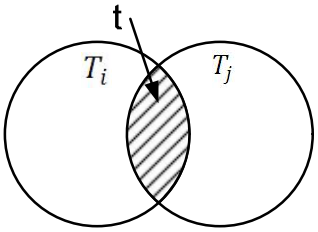
$$HasConc(av\langle s_i\rangle, av\langle s_j\rangle) \rightarrow (T_i \cap T_j \neq \emptyset)$$

Suatu konkurensi dapat dipandang sebagai suatu kejadian. Oleh karenanya suatu konkurensi dapat dipandang sebagai suatu situasi sehingga memiliki state konteks dan state mobilitas konteks. Ada dua jenis state konteks terkait konkurensi, yaitu *state konteks irisan* dan *state konteks gabungan*. *State konteks irisan* merupakan state konteks yang merepresentasikan apakah avatar berada di dalam konkurensi atau tidak. Saat dimana avatar *av* sedang dalam konteks "kuliah" sekaligus dalam konteks

 ISSN: 1411-3201

"mengajar" (t_2 sampai t_3 pada Gambar 7) merupakan contoh saat dimana state konteks irisan terjadi. State konteks irisan antara dua konteks ($av\langle s_i\rangle$ dan $av\langle s_j\rangle$) terjadi (bernilai *true*) ketika avatar berada di dalam *t* yang merupakan irisan kedua state konteks primitif penyusunnya (Gambar 8). State konteks irisan dua konteks state antara $av\langle s_i\rangle$ dan $av\langle s_j\rangle$ dinotasikan dengan $av\langle s_i \cap s_j\rangle$ dan didefinisikan sebagai berikut :

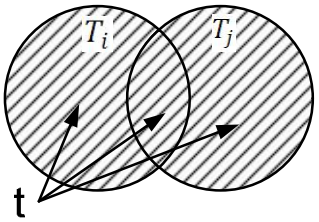
$$av\langle s_i \cap s_j\rangle = \begin{cases} true & jika\ t \in T_i \cap T_j \\ false & selainnya \end{cases}$$



Gambar 8. Irisan waktu kejadian konkuren

State konteks gabungan merupakan state konteks yang merepresentasikan saat dimana avatar mulai berada di dalam state konteks penyusun konkurensi dari yang yang paling awal sampai paling akhir. Saat dimana avatar *av* mulai berada dalam konteks "kuliah" sampai berakhirnya konteks "mengajar" (t_1 sampai t_4 pada Gambar 7) merupakan merupakan contoh saat dimana state konteks gabungan terjadi. State konteks gabungan antara dua konteks ($av\langle s_i\rangle$ dan $av\langle s_j\rangle$) terjadi (bernilai *true*) ketika avatar berada di dalam *t* yang merupakan gabungan kedua state konteks primitif penyusunnya (Gambar 9). State konteks gabungan dua konteks state antara $av\langle s_i\rangle$ dan $av\langle s_j\rangle$ dinotasikan dengan $av\langle s_i \cup s_j\rangle$ didefinisikan sebagai berikut :

$$av\langle s_i \cup s_j\rangle = \begin{cases} true & jika\ t \in T_i \cup T_j \\ false & selainnya \end{cases}$$



Gambar 9. Gabungan waktu kejadian konkuren

Model konteks konkurensi irisan dan gabungan dari dua buah konteks dapat digeneralisir untuk diterapkan pada konteks berjumlah banyak. State konteks irisan dari *m* kejadian dapat didefinisikan dalam notasi sebagai berikut :

$$av\langle s_1 \cap s_2 \cap \ldots \cap s_m\rangle =$$

$$av\langle \bigcap_1^m s_i\rangle = \begin{cases} true & jika\ t \in (T_1 \cap T_2 \cap \ldots \cap T_m) \\ false & selainnya \end{cases}$$

Sementara state konteks gabungan dari *n* kejadian juga dapat didefinisikan sebagai berikut :

$$av\langle s_1 \cup s_2 \cup \ldots \cup s_n\rangle =$$
$$av\langle \bigcup_1^n s_i\rangle = \begin{cases} true & jika\ t \in (T_1 \cup T_2 \cup \ldots \cup T_n) \\ false & selainnya \end{cases}$$

State konteks primitif dapat dinyatakan dalam bentuk pasangan nilai konteks dan suatu pencacah untuk memberikan identitas yang unik terhadap kejadian yang mungkin berulang. Oleh karenanya state konteks irisan $av\langle \bigcap_{i=1}^m s_i\rangle$ dan state konteks gabungan $av\langle \bigcup_1^n s_i\rangle$ juga dapat diberikan identifier dalam bentuk berturut-turut : $av\langle (\bigcap_1^m x_i, n_j)\rangle$ dan $av\langle (\bigcup_1^n x_i, n_j)\rangle$. State mobilitas dari konteks irisan imodelkan sebagai berikut: **(4.9)**

$$av\langle \diamond\ (\bigcap_1^m x_i, n_i)\rangle = \begin{cases} true & jika\ t < t_1 \\ false & selainnya \end{cases}$$
$$av\langle (\bigcap_1^m x_i, n_i)\rangle = \begin{cases} true & jika\ t_1 \leq t \leq t_m \\ false & selainnya \end{cases}$$
$$av\langle (\bigcap_1^m x_i, n_i)\ \diamond\rangle = \begin{cases} true & jika\ t > t_m \\ false & selainnya \end{cases}$$

Sementara untuk state mobilitas untuk state konteks gabungan dapat dituliskan sebagai berikut :

$$av\langle \diamond\ (\bigcup_1^n x_i, n_j)\rangle =$$
$$\begin{cases} true & jika\ t < t_1 \\ false & selainnya \end{cases}$$
$$av\langle (\bigcup_1^n x_i, n_j)\rangle =$$
$$\begin{cases} true & jika\ t_1 \leq t \leq t_n \\ false & selainnya \end{cases}$$
$$av\langle (\bigcup_1^n x_i, n_j)\ \diamond\rangle = \begin{cases} true & jika\ t > t_n \\ false & selainnya \end{cases}$$

Avatar, Fungsi dan Context Awareness **(4.10)**

Avatar adalah entitas obyek virtual yamg melakukan interaksi di dalam lingkungan virtual. Obyek virtual secara mendasar dimodelkan dalam bahasa *Ambient Calculus* berikut ini.

av_k	::= 0	null
	| M	nama obyek
	| $av_k[F_1, F_2, \ldots, F_n]$	keterkandungan fungsi

Suatu obyek avatar berikut fungsi yang dikandungnya menjadi suatu kesatuan avatar yang utuh. Fungsi yang dimiliki avatar merepresentasikan kemampuan dari avatar untuk dapat melakukan aksi.

Fungsi merupakan suatu potensi avatar untuk melakukan aksi. Meskipun memiliki fungsionalitas, suatu entitas komputasi adalah konstruksi obyek pasif yang tidak memiliki inisiatif untuk mengaktifkan fungsionalitasnya tersebut. Suatu

 ISSN: 1411-3201

fungsionalitas bekerja berdasarkan suatu faktor eksternal.

Dalam entitas avatar, aktivasi suatu fungsional berkaitan dengan konteks yang dialami oleh avatar tersebut. Suatu fungsi akan aktif ketika suatu pola situasi tertentu terjadi pada avatar (reaktif). Agar mekanisme reaktif tersebut dapat berjalan dengan baik, avatar perlu memiliki *awareness* atau kepekaan terhadap konteks yang terlibat dalam skema reaktivitas tersebut (*context awareness*). Avatar perlu mengamati setiap perubahan pada state konteks tersebut sehingga dapat menentukan untuk mengeksekusi fungsi yang sesuai ketika situasi konteks yang memenuhi terjadi. Misal : avatar dosen *av* melakukan aksi *Presentasi*, ketika berada dalam konteks "di dalam kuliah". Dalam kasus tersebut *Presentasi* merupakan suatu fungsi yang hanya aktif ketika konteks "di dalam kuliah" terjadi pada avatar tersebut. Menggunakan Ambien Calculus, maka kasus tersebut dapat dimodelkan sebagai suatu mekanisme reaktifitas sebagai berikut.

$$av[Presentasi]\langle"di\ dalam\ kuliah",*\rangle \\ \rightarrow Dosen[\overline{Presentasi}]\langle"di\ dalam\ kuliah",*\rangle$$

Mekanisme aktivasi yang lebih kompleks ditunjukkan dalam beberapa notasi di bawah ini. Sebagai contoh, avatar *mhs* adalah avatar seorang mahasiswa yang sedang dalam proses merevisi proposal tesisnya. Untuk skenario tersebut, *mhs* memiliki kemampuan untuk menyelesaikan revisi proposal (ditunjukkan dengan fungsi *finish_rev*) dan mensubmit proposal (ditunjukkan oleh fungsi *submit*). Aktivasi dari kedua fungsi tersebut dipicu oleh beberapa state mobilitas konteks yang terjadi pada *mhs* berikut ini.

$$mhs[finish\_rev, submit]\langle \diamond\ (rev\_finished,*)\ \rangle \\ \rightarrow mhs[\overline{finish\_rev}, submit]\langle \diamond\ (rev\_finished,*)\rangle$$

$$mhs[finish\_rev, submit]\langle rev\_finished \cap \\ \underline{\qquad\qquad} submit\_period,*\rangle \\ \rightarrow mhs[finish\_rev, \overline{submit}]\langle rev\_finished \cap \\ submit\_period,*\rangle$$

Notasi tersebut menjelaskan bahwa sebelum *mhs* menyelesaikan revisi proposal tersebut (ditunjukkan oleh "$\diamond\ (rev\_finished,*)$"), maka *mhs* akan mengaktifkan fungsi *finish_rev* yaitu fungsi untuk menyelesaikan revisi proposal.

Pada notasi berikutnya, fungsi untuk melakukan submit proposal ("*submit*") dilakukan ketika revisi proposal tersebut telah selesai (konteks $rev\_finished$) dan *mhs* sudah dalam periode untuk mensubmit proposal (konteks $submit\_period$). Fungsi *submit* dilakukan tepat ketika dua konteks tersebut terjadi secara bersamaan (terjadi konkurensi), sehingga hal ini juga menjelaskan bahwa *mhs* tidak akan mensubmit proposal ketika periode submit belum tiba walaupun proposal telah selesai direvisi. Begitu pula proposal tidak akan disubmit ketika periode submit telah tiba namun revisi proposal belum selesai dilakukan.

Kesimpulan

Berdasarkan uraian bab-bab sebelumnya, maka dapat diambil kesimpulan sebagai berikut.

1. Konteks telah dimodelkan untuk memberikan fundamen dalam membuat sistem interaksi avatar yang berbasis konteks dengan pendekatan yang generik untuk menyediakan dukungan yang fleksibel terhadap dinamika dan ragam konteks. Sensitivitas avatar terhadap konteks dapat ditunjukkan dari kerangka kerja dengan adanya mekanisme reaktif sebagai pemicu eksekusi dari suatu fungsionalitas dari avatar.

2. Kerangka kerja *Mocha* sebagai telah dikembangkan untuk dapat memodelkan situasi konteks konkuren sebagai suatu situasi konteks tersendiri yang dapat digunakan dalam mekanisme reaktif eksekusi fungsional avatar.

Daftar Pustaka

[1] Bailenson, J.N., dan Blascovich, J, *Avatars. Encyclopedia of Human-Computer Interaction*, Vol 1, pp. 64-6, Great Barrington, MA: Berkshire Publishing Group, 2004

[2] Clark, H.H., Brennan, S.E, *Grounding in Communication*, in L.B. Resnick, J.M. Levine and S.D. Teasley (eds.), Perspectives on Socially Shared Cognition, American Psychological Association, pp. 127-149, 1991

[3] Crabtree, Andy, Rodden, Tom, dan Mariani, John, Designing *Virtual* Environments To Support Cooperation In The real World. *Virtual reality. 6 (2). Pp. 63-74. ISSN 1359-433*, 2002

[4] Nugroho, L.E., *A Context-Based Approach for Mobile Application Development.* School of Computer Science and Software Engineering. Monash University, 2001

[5] Tripathi, A.R., Kulkarni, D., Ahmed, T., *A Specification model for context-based collaborative application*, Pervasive and Mobile Computing 1(1): 21-42, 2005

[6] Wilcox, C.D., dan Roman, G-C., *Reasoning about places, times, and actions in the presence of mobility*, IEEE Transactions on Software Engineering, 22(4):225-247, April 1996

(4.23)