MODEL BERBASIS ONTOLOGI DAN ALGORITMA KOLONI SEMUT UNTUK PENCARIAN JALUR TERPENDEK PADA *CONTEXT-AWARE INDOOR NAVIGATION*

Yisti Vita Via[1], Salamun Rohman Nudin[2]
[1] Teknik Informatika, Fakultas Teknologi Informasi, Universitas Pembangunan Nasional "Veteran"
Jalan Raya Rungkut Madya, Gunung Anyar, Surabaya, 60294, Indonesia
[2] Teknik Elektro, Fakultas Teknik, Universitas Negeri Surabaya
Postal Address
email : yistivita@gmail.com[1], your2@email2 [2]

Abstrak: Pada sistem navigasi ruangan yang masih tradisional, posisi awal dan akhir dari jalur navigasi harus terdefinisikan dahulu sebelum proses navigasi berlangsung. Penelitian ini mengangkat konteks ke dalam permasalahan sistem navigasi ruangan untuk memberikan layanan kepada pengguna dalam mencari posisi target yang berpindah-pindah dan belum diketahui sebelumnya. Teknologi ontologi dan *reasoning* digunakan untuk memberikan keputusan penentuan jalur proses navigasi kepada pengguna. Sedangkan algoritma Koloni Semut digunakan untuk optimasi sistem navigasi dengan melakukan pencarian jalur navigasi yang terpendek. Hasil sistem ini mampu menunjukkan konsep pengetahuan, hubungan antar konsep, dan *inference rules*, serta memberikan skenario tahapan jalur navigasi terpendek kepada pengguna.

Keywords: ontologi, navigasi ruangan, *reasoning*, algoritma Koloni Semut, *inference rules*.

1. PENDAHULUAN

Di dalam kehidupan nyata terdapat beberapa permasalahan yang perlu diselesaikan agar dapat memberikan layanan sistem navigasi yang lebih baik. Misalnya, di kampus dan gedung perkantoran, terkadang beberapa pintu dibuka setiap hari dalam seminggu tetapi ditutup pada malam hari dan akhir minggu. Hal ini berarti di waktu yang berbeda, ada rute berbeda yang harus dilalui di antara lokasi-lokasi yang berbeda pula. Keadaan lainnya terjadi di lingkungan akademik. Seorang profesor memberikan kuliah atau rapat pada waktu dan tempat yang sudah dijadwalkan. Sehingga seseorang yang ingin bertemu dengannya harus mengetahui lokasi beliau berada berdasarkan jadwal kegiatan yang bersesuaian. Oleh karena itu, sebuah sistem navigasi yang dapat berhubungan dengan konteks yang berbeda diperlukan untuk menyelesaikan permasalahan ini.

Beberapa sistem navigasi telah dibangun untuk memberikan petunjuk di dalam ruangan. Banyak penelitian [1, 2] fokus pada interaksi *Human Computer Interaction* (HCI) untuk memberikan pengguna kondisi yang berbeda dari paradigma navigasi yang lebih baik. Penelitian lainnya [3, 4] menggunakan model dan struktur yang berbeda untuk mencakup informasi lokasi dan konteks. Meskipun demikian, ada dua kekurangan dalam kinerja penelitian-penelitian ini yaitu pengguna harus mengetahui sebelumnya lokasi target dengan jelas dan keadaan lingkungan sistem navigasi masih bersifat statis.

Sebuah penelitian [5] *Context-Aware Indoor Navigation* (CAIN) dapat menangani permasalahan ini yaitu untuk diterapkan pada sistem navigasi dengan keadaan lingkungan yang dinamis dan lokasi target yang belum diketahui sebelumnya. Dengan menggunakan informasi waktu dan kegiatan, posisi target dapat diketahui dengan menemukan lokasi kegiatan yang sedang dihadiri target. Proses ini menggunakan ontologi dan *reasoning* untuk memodelkan konteks, menemukan jalur, dan memberikan jalur navigasi kepada penggunanya.

Pada paper ini, teknologi CAIN akan dijelaskan pada Bagian 2. Sedangkan algoritma Koloni Semut akan dijelaskan pada Bagian 3. Model representasi pengetahuan dan sistem arsitektur diperkenalkan di bagian 4 dan 5. Bagian 6 membahas skenario uji coba dan pembahasan untuk menggambarkan proses dari sistem. Dan kesimpulan diberikan di bagian 7.

2. CONTEXT-AWARE INDOOR NAVIGATION SYSTEM

Banyak peneliti telah bekerja lama pada *intelligent routing*, navigasi, dan *guidance systems*. Cyberguide [8] mengembangkan sebuah *location-*

aware guidance application yang mendukung *tour indoor* dan *outdoor* meliputi *positioning*, peta, dan komponen informasi di kampus. NAVIO [9] memfokuskan pada integrasi dan penyatuan pejalan kaki berjalan antara lokasi *indoor* dan *outdoor*. Penelitian ini hanya bisa digunakan pada lingkungan yang statis sehingga tidak dapat memutuskan tujuan jalur navigasi yang tidak diketahui.

Sekarang ini, peneliti mengadopsi ide dari konteks *awareness* untuk membuat layanan navigasi yang lebih baik. OntoNav [1] memberikan suatu paradigma navigasi *user-centric* untuk lingkungan *indoor* berdasarkan pada kemampuan dan keterbatasan fisikal pengguna seperti pengguna normal, pengguna cacat atau pengguna dengan membawa banyak barang di tangannya. *Indoor wayfinding* untuk pengguna cacat telah dilakukan pada profil, konteks waktu dan lokasi, serta mengembangkan sebuah *prototype interface* pengguna. Penelitian ini fokus pada desain *interface* komputer manusia dan tidak memutuskan model permasalahan navigasi.

Untuk memodelkan informasi konteks dan memberikan layanan navigasi, peneliti menggunakan metode dan representasi yang berbeda. CoINS [3] menggunakan *convex hull* dan *quadtree* untuk merepresentasikan lokasi *indoor*, dan menggunakan ide jalur berdekatan untuk mengembangkan layanan pemberi petunjuk. Hsieh et al. [10] mengajukan model penghitungan jalur *hybrid* untuk menyelesaikan permasalahan navigasi *indoor*. Penelitian ini mencoba memperbaiki dan menyelesaikan algoritma *pathfinding* dengan menggunakan model navigasi mereka, yang mana tisak berurusan dengan penanganan konteks.

Context-aware mengambil informasi kontektual ke dalam laporan dan memberikan layanan untuk memenuhi kebutuhan pengguna. Selama beberapa tahun lalu, sejumlah sistem *context-aware* telah dikembangkan untuk mendukung penhitungan pervasif dan berkenaan dengan lingkungan intelijen seperti sistem lokasi Active Badge[11], ParcTab[12], dan Context Tollkit[13]. Sistem ini menggunakan bermacam-macam sensor dan peralatan untuk memberikan layanan *location-aware* tetapi tidak mempertimbangkan persoalan penalaran konteks.

Banyak context-aware systems berkonsentrasi pada layanan *location aware* seperti Semantic geoStu. RFC 2445[5] mendefinisikan format iCalendar dan aplikasi penjadwalan, yang membantu pengguna untuk membuat aktivitas pribadi. Google Calendar[6] adalah kalender berbasis web yang sangat terkenal mendukung standar iCalendar dan pengguna dapat berbagi aktivitas pribadi dengan pengguna yang lain. Aktivitas ini terhubung dengan orang lain, waktu, dan lokasi. konsekuensinya, konten jadwal ini dapat membantu menemukan lokasi seseorang pada waktu tertentu.

Pada CAIN [5], ingkungan yang berada di dalam ruangan dapat direpresentasikan dalam $G = (V, E)$, dimana himpunan node $V = \{v0, v1, v2, \dots, vn\}$ merupakan kumpulan ruangan dan himpunan edge $E = \{(vi, vj) | (vi \neq vj), vi, vj \in V\}$ merupakan kumpulan segmen jalur yang menghubungkan dua ruangan.

Dimisalkan posisi awal *s* dan posisi akhir *d*, layanan navigasi ruangan memberikan jalur dari tempat *s* ke *d* dan setiap potongan jalur dihubungkan dengan anak panah. Didefinisikan sebuah jalur $r_1 \overrightarrow{o_1}\ r_2 \overrightarrow{o_2}\ \dots\ \overrightarrow{o_{d-1}}\ r_{1d}$, dimana $r_1, r_2, \dots, r_d \in V, s = r_1, d = r_d,\ (r_1, r_2),\ (r_2, r_3),$ $\dots, (r_{d-1}, r_d) \in E$, dan arah panah $o_1, o_2, \dots, o_{d-1}$.

Pada penelitian ini ditambahkan kemampuan context-aware pada layanan navigasi indoor yang asli. Ketika pengguna berlokasi di gedung, posisi awal *s* dapat disebut sebagai posisi saat ini pengguna. Navigasi indoor membantu pengunjung baru dalam sebuah gedung untuk menemui target tujuannya, yang bisa berupa seseorang maupun lokasi ruangan. Jika target adalah seseorang, lokasinya bisa dilihat dari jadwal kegiatannya. Sebagai contoh seorang profesor mungkin bergerak ke ruangannya, laboratorium, ruang kelas, atau ruang seminar. Untuk itu, posisi akhir *d* untuk CAIN, tidak harus lokasi, tetapi disimpulkan dari konteks target. Pada intinya CAIN ini membantu layanan navigasi dari lokasi user saat itu ke lokasi targetnya. Pada waktu yang berbeda, lokasi target ini bisa berubah sesuai dengan konteks.

3. ALGORITMA KOLONI SEMUT

Algoritma Koloni Semut pertama kali dikenalkan oleh Dorige et al. [17] untuk menyelesaikan *Traveling Salesman Problem* (TSP). Algoritma ini diadopsi dari perilaku koloni semut yang dikenal sebagai sistem semut. Secara alamiah koloni semut mampu menemukan rute terpendek dalam perjalanan dari sarang ke tempat-tempat sumber makanan. Koloni semut dapat menemukan rute terpendek antara sarang dan sumber makanan berdasarkan jejak kaki pada lintasan yang telah dilalui. Semakin banyak semut yang melalui suatu lintasan, maka akan semakin jelas bekas jejak kakinya. Hal ini akan menyebabkan lintasan yang dilalui semut dalam jumlah sedikit, semakin lama akan semakin berkurang kepadatan semut yang melewatinya, atau bahkan akan tidak dilewati sama sekali. Sebaliknya lintasan yang dilalui semut dalam

jumlah banyak, semakin lama akan semakin bertambah kepadatan semut yang melewatinya, atau bahkan semua semut akan melalui lintasan tersebut.

Untuk menyelesaikan permasalahan dengan algoritma Koloni Semut, dalam banyak kasus, masalah harus didefinisikan dan direpresentasikan dengan *graph*. Setelah itu semut mulai menghasilkan solusi dalam *graph* dan mereka menggunakan jejak feromon dan informasi *heuristic* sebagai petunjuk. Informasi *heuristic* ini merupakan ukuran prefensi untuk bergerak dari state S_i ke S_j dan jejak feromon adalah jumlah feromon yang disimpan oleh semut pada langkah sebelumnya yang menunjukkan sifat pembelajaran dari pergerakan posisi S_i ke S_j. Sebagai pertimbangan solusi yang telah ditemukan, pada setiap langkah semut mengupdate jumlah jejak feromon. Pada langkah selanjutnya, titik dengan jumlah jejak feromon yang banyak akan lebih disukai untuk dipilih semut.

Pada beberapa tahun terakhir banyak penelitian yang telah mengembangkan berbagai algoritma Koloni Semut untuk permasalahan yang beragam seperti *vehicle routing problem*, *traveling salesman problem*, *producting scheduling*, *sequential ordering problem*, *telecomunication routing*, dan lainnya. Beberapa algoritma Koloni Semut ini diajukan oleh banyak peneliti dengan prosedur update jejak feromon, penguapan, dan aturan transisi yang berbeda. Sistem semut pertama kali dikembangkan oleh Dorige et al. [17] dan diterapkan untuk menyelesaikan *classic Traveling Salesman Problem*. Enam tahun kemudian Dorigo dan Gambardella [18] mengenalkan algoritma yang lain dengan performansi yang lebih baik dan disebut algoritma Koloni Semut. Algoritma Koloni semut ini menggunakan prosedur yang berbada di update jejak feromon lokal dan global sebaik aturan transisi. Terdapat beberapa versi dari algoritma Koloni Semut seperti *Elitish Ant Sistem* [19], *rank-based Ant System* [19, 20], dan *Min-Max Ant System* [19, 21].

Dalam algoritma Koloni Semut, diperlukan beberapa variabel dan prosedur untuk menentukan jalur terpendek. Tahapan prosedurnya sebagaimana ditunjukkan oleh flowchart pada Gambar 1.

Masing-masing semut memodifikasi lingkungannya dengan dua cara yang berbeda.

- Update feromon lokal. Semut bergerak dari satu titik ke titik yang lain dengan mengupdate sejumlah feromon pada setiap tepi (*edge*) dengan mengikuti persamaan (1).

$$\tau_{ij}(t) = (1-\rho) + \rho\tau_o \ldots\ldots\ldots\ldots\ldots\ldots (1),$$

dimana ρ merupakan konstanta evaporasi (penguapan). Nilai τ_o merupakan inisial nilai dari jejak feromon. Nilai ini diperoleh dengan menggunakan rumus $\tau_o = (nL_{nn})^{-1}$ dimana n merupakan jumlah kota dan L_{nn} panjang *tour* yang diperoleh dengan satu *heuristic* atau metode pelacakan *nears neighborhood*.

- Update feromon global. Ketika semua semut telah melaksanakan satu *tour* secara lengkap, dan semut tersebut menemukan rute terpendek. Maka tepian-tepian yang dilewati oleh semut tersebut akan di-update feromonnya dengan menggunakan persamaan (2).

$$\tau_{ij}(t) = (1-\rho)\tau_{ij} + \frac{\rho}{L^+} \ldots\ldots\ldots\ldots\ldots (2),$$

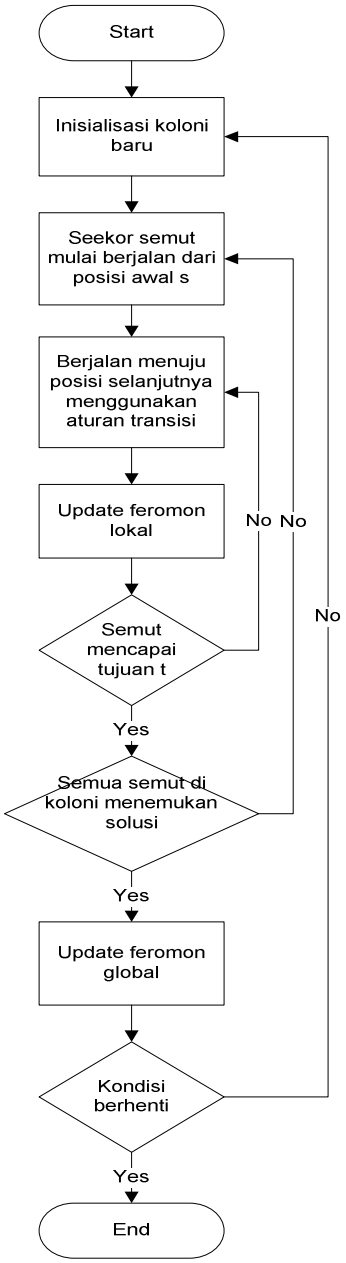
dimana L^+ merupakan panjang dari *best tour* yang ditemukan oleh satu semut dari beberapa semut yang bekerja.

4. MODEL ONTOLOGI

Ontologi adalah spesifikasi secara konsep yang formal dan eksplisit terdiri dari daftar terminologi yang terbatas dan hubungan antara terminologi-terminologi itu [14]. Ontologi dapat digunakan sebagai model data yang merepresentasikan sebuah domain dan memberikan pengetahuan untuk reasoning tentang objek dan hubungannya dalam domain. Gambar 1 menggambarkan ontologi sebagai representasi dari konsep biasanya tentang navigasi lokasi di dalam ruangan dan sesuai untuk pencarian jalur.

Informasi konteks dikumpulkan dari *class real-word* seperti *Event*, *Sensor*, *Time*, *Location*, dan *Person*. Paper ini mengangkat konteks ini dan memberikan layanan navigasi yang sadar konteks yang direpresentasikan dalam konsep navigasi.

Class hierarchy merepresentasikan sebuah hubungan subkelas; sebuah arah panah dari suatu *subclass* ke *superclass* yang lain. Dapat dilihat pada Gambar 2, *PlaceIndoor class* adalah konsep dari lokasi, yang mana *is-a* merupakan representasi hubungan *subclass*.

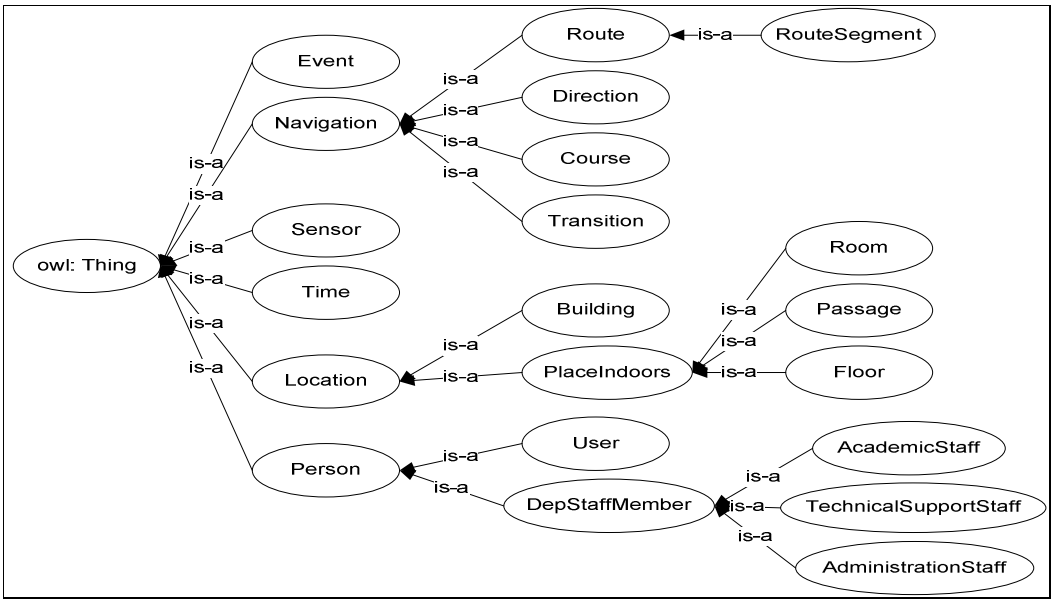


Gambar 1. Flowchart algoritma Koloni Semut.

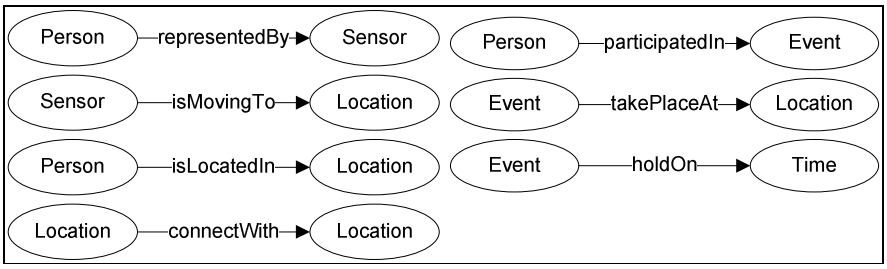
Pada Gambar 2, konsep *Route* dimulai dari posisi awal *s* ke posisi tujuan akhir *d*, dimana *s* dan *d* adalah instan dari *PlaceIndoor*. Dimisalkan bahwa jalur $r_1\ \overrightarrow{o_1}\ r_2\ \overrightarrow{o_2}\ \ldots\ \overrightarrow{o_{d-1}}\ r_{1d}$ terdiri dari beberapa potongan jalur yang terhubung yang direpresentasikan dalam *RouteSegment class*. Untuk itu, titik (r_1, r_2), (r_2, r_3), …, (r_{d-1}, r_d) dan panah $o_1, o_2, \ldots, o_{d-1}$ adalah instan dari *RouteSegment* dan *Direction* tentunya. Jalur merupakan titik yang terhubung berurutan, hubungan *connectWith* menggambarkan hubungan antara dua titik yang diminta. Pada Gambar 3 menunjukkan *connectWith* sebagai hubungan antar dua instan *Location*.

Posisi awal *s* dari jalur navigasi didapatkan dari posisi pengguna, yang mana bisa didapatkan dari lokasi sensor [15]. *Person* dalam hal ini misalnya John dapat diidentifikasi oleh *Sensor* sensor1, yang dijelaskan oleh hubungan *representedBy*. Properti *isMovingTo* menggambarkan perpindahan John yaitu ketika John bergerak ke satu lokasi, sensor yang dia bawa akan mendeteksi pergerakannya. Properti *isLocatedIn* merepresentasikan bahwa seseorang itu berada di suatu tempat.

Jika target jalur navigasi adalah seseorang, layanan *context-aware* harus mengecek jadwal kegiatan target tersebut. Pada Gambar 3, properti *participatedIn* menggambarkan bahwa instan dari *Person is participated in an Event*. Instan dari *Event* akan mengambil posisi pada *Location* dengan *Time* yang berkaitan. Hal ini menyatakan bahwa properti *takePlaceAt* dan *HoldOn* merepresentasikan hubungan kapan dan dimana dari sebuah kegiatan tentunya.



Gambar 2. Ontologi Navigasi Ruangan.



Gambar 3. Hubungan antar kelas ontologi.

5. ARSITEKTUR SISTEM

Arsitektur dari sistem ini ditunjukkan pada Gambar 4. Sistem terbagi menjadi beberapa bagian blok yang menjalankan fungsi masing-masing. Alur kerja dari sistem ini dimulai dari *Graphical User Interface* (GUI) yang mengirimkan permintaan dari pengguna kepada Context Aware Service Platform. Kemudian Context Collection Agents mengirimkan konteks-konteks pendukung dari Context Resources untuk digunakan oleh Context Aware Services Platform dalam mencari jalur-jalur yang bisa dilalui. Selanjutnya jalur-jalur ini akan diseleksi sehingga didapatkan satu jalur terpendek yang merupakan keluaran dari sistem ini. Jalur terpendek ini kemudian disajikan kepada pengguna melalui GUI. Semua agen-agen ini menggunakan pesan untuk berkomunikasi satu sama lain.

GUI pada arsitektur sistem ini merupakan penghubung utama antara pengguna dan sistem. GUI menerima permintaan dari pengguna untuk dikirimkan ke sistem. Dan sebaliknya GUI juga menerima instruksi navigasi dari sistem untuk diberikan kepada pengguna.

Bagian lain dari sistem ini adalah *Context Resources*. Bagian ini dapat diperoleh dari *software* atau *hardware* lain. Pada kasus ini, konteks yang digunakan dapat diperoleh dari kalender, profil pengguna, dan sensor lokasi. Lokasi pengguna didapatkan dari sensor lokasi, profil pengguna dapat diketahui dari konteks profil, dan jadwal kegiatan target dapat diambil dari kalender. Beberapa konteks ini akan dikirimkan ke *Context Collection Agents*.

Selanjutnya *Context Collection Agents* akan mengkonversi konteks-konteks yang telah diterima dari *Context Resources* menjadi representasi semantik dan kemudian dikirimkan ke bagian *Context Aware Service Platform*.

Context Aware Service Platform merupakan inti dari arsitektur sistem ini, terbagi menjadi

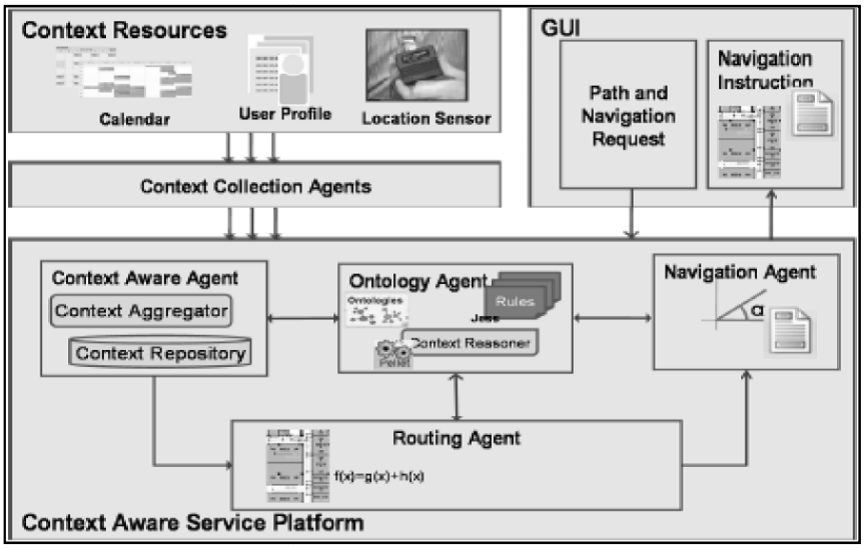
beberapa agen lagi yaitu: *context aware agent*, *ontology agent*, *route agent*, dan *navigation agent*.

- *Context Aware Agent* menerima informasi kontekstual dan menjaga konsistensi konteks. Agen ini terdiri dari *Context Aggregator* dan *Context repository. Context Aggregator* mengumpulkan konteks dari *Context Collection Agents* dan *Ontology Agent*. Sedangkan *Context Repository* menyimpan semua konteks-konteks ini.
- *Ontology Agent* menggunakan ontologi pada Gambar 2 untuk mengkonversi lokasi sensor ke dalam lokasi ruangan. Dalam penambahannya, *reasoner* konteks mengkombinasikan *rule-based engine* Jess[7] dan DL *reasoner* Pellet untuk melakukan *query matching* dan mengambil kesimpulan *high-level context*. Sebagai contoh, untuk mencari lokasi seseorang, sebuah aturan akan dibangkitkan untuk mencocokkan waktu dan jadwal untuk mendapatkan orang tersebut sedang melakukan aktivitas apa dan dimana.
- *Routing Agent* membangkitkan beberapa jalur dari posisi awal hingga posisi akhir. Jalur-jalur ini yang kemudian akan diseleksi untuk didapatkan jalur yang terpendek dengan menggunakan algoritma Koloni Semut. Jalur ini terbagi menjadi potongan-potongan jalur yang nanti akan dikirimkan ke bagian *Navigation Agent*.
- *Navigation Agent* bertugas mengirimkan instruksi navigasi pada GUI. Ketika user sedang bergerak di suatu tempat, *Navigation Agent* akan menghitung orientasi dan arah yang bersesuaian berdasarkan potongan-potongan jalur yang diberikan berkelanjutan.

6. UJI COBA DAN PEMBAHASAN

Uji coba dilakukan untuk memberikan gambaran proses secara detail bagaimana sistem menggunakan ontologi dan algoritma Koloni Semut dalam memberikan layanan CAIN. Pada uji coba ini diberikan sebuah contoh kasus pencarian target.

Dr. Agus adalah seorang dosen di jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya. Beliau sangat sibuk dalam memberikan kuliah, rapat, dan diskusi penelitian dengan mahasiswanya. Ketika Yisti ingin bertemu dengan Dr. Agus, dia menggunakan layanan CAIN untuk memberikan petunjuk dalam mencari keberadaan Dr. Agus.



Gambar 4. Arsitektur Sistem.

Dalam membantu Yisti untuk menemukan Dr. Agus, layanan CAIN akan melakukan beberapa tahapan proses seperti ditunjukkan pada Gambar 5.

Gambar 5. Tahapan proses pada sistem navigasi.

6.1 Menentukan Posisi Awal

Sensor pendeteksi lokasi dapat mendeteksi pergerakan pengguna, dalam hal ini Yisti. Aktifitas ini melibatkan *Context Collection Agent*, *Context Aware Agent*, dan *Ontology Agent*. Konteks dan pengetahuan direpresentasikan dalam RDF-triple dan format objek. Subjek merupakan *resource* yang diberi nama URI dengan identitas optional. Predikat merupakan properti *resource*.

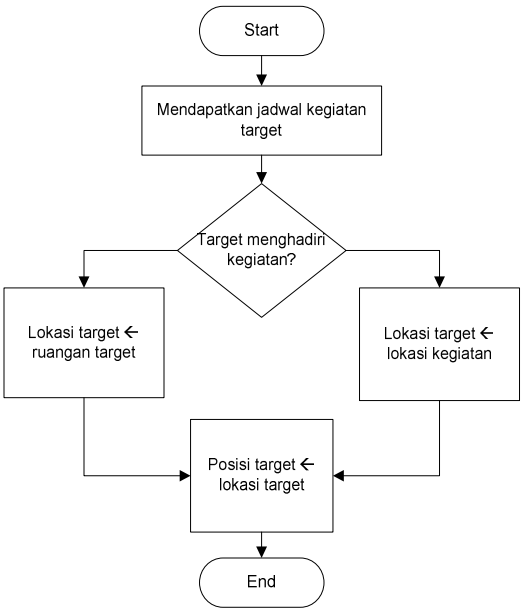
Ketika Yisti bergerak ke suatu lokasi, sensor akan mengirimkan lokasinya ke *Context Collection Agent*. Dimisalkan *sensor1* bergerak ke *loc1*, dimana *sensor1* dan *loc1* adalah instan dari *Sensor* dan *Location*.

Dalam *Context Aware Agent*, *Context Aggregator* akan menerima dan menyimpan konteks baru ini pada *Context Repository*. *Context Collection Agent* mengirimkan data sensor ke *Context Aware Agent* dan *Ontology Agent*.

Ketika *Ontology Agent* menerima baris data sensor, sebuah *query* akan mencocokkan bahwa Yisti dikenali oleh *sensor1*. *Ontology Agent* yang terdiri dari *context reasoner*, menggunakan aturan-aturan untuk menyimpulkan konteks baru. Aturan dalam *ruled based system* ditunjukkan dalam pernyataan IF-THEN. *Context Reasoner* membuat aturan dan menggunakannya untuk menyimpulkan konteks baru. Demikian hingga semua kondisi dicocokkan sehingga context reasoner akan menyimpulkan bahwa "*Yisti* berlokasi di *loc1*".

6.2 Menentukan Posisi Target

Posisi target ini merupakan lokasi dari Dr. Agus. *Ontology Agent* akan mencari lokasi beliau dari kalender kegiatannya. Gambar 6 menunjukkan bahwa jika Dr. Agus sedang menghadiri kegiatan maka beliau akan berada di lokasi di mana sudah dituliskan di kalender kegiatannya. Hubungan antara kegiatan dan lokasi adalah hubungan *takePlaceAt*.



Gambar 6. Tahapan proses penentuan node tujuan.

Jika lokasi Dr. Agus sudah ditemukan, maka lokasi ini yang akan digunakan sebagai posisi target ujuan pada jalur navigasi.

6.3 Membangkitkan Jalur Navigasi

Routing Agent menerima posisi awal *s* dan posisi tujuan *d* dari *Context Aware Agent* dan *Ontology Agent*. Algoritma *routing* yang dipakai dalam mencari jalur navigasi adalah LRTA* [16] dimana jarak dari posisi awal ke posisi sekarang adalah harga nyata dan jarak garis lurus dari posisi sekarang ke posisi terakhir adalah fungsi *heuristic*.

Seperti pada *Breadth First Search* (BFS), *Route Agent* akan mencari titik-titik lokasi yang terhubung pada jalur. Karena keamanan, setelah jam kantor, bangunan kantor akan menutup beberapa pintu masuk dan membuka satu pintu untuk akses bebas. Informasi kontrol keamanan ini dapat didefinisikan sebagai atribut bangunan, ruangan, dan lorong. Dengan menerapkan pendekatan yang sama seperti Gambar 6, sistem menggunakan waktu yang tersedia untuk menentukan mana titik yang bisa terhubung dan tidak.

Pada Gambar 3 didefinisikan *connectWith* sebagai hubungan antara lokasi. Sebuah atribut potongan jalur *class RouteSegment* pada Gambar 2 menggambarkan waktu *edges* yang tersedia. Sehingga *Routing Agent* akan menghasilkan jalur dari lokasi *s* ke lokasi *d* dan setiap *edge* jalur ini dihubungkan.

6.4 Menentukan Jalur Navigasi Terpendek

Pada tahap ini algoritma Koloni Semut digunakan untuk mencari jalur terpendek di antara jalur-jalur yang telah dibangkitkan pada tahap sebelumnya. Jalur terpendek ini dianggap optimal dalam menemukan target karena akan dapat mengurangi frekuensi perpindahan target sebelum target ditemukan.

Jalur terpendek ini akan dikirimkan *Routing Agent* ke *Navigation Agent* untuk diteruskan ke GUI.

6.5 Menampilkan Tahapan Jalur Navigasi

Navigation Agent menggunakan ketentuan pada Gambar 2 untuk memperhitungkan petunjuk navigasi dan mengirimkan instruksi pada GUI berdasarkan tahapan-tahapan berurutan dari jalur navigasi terpendek yang ditentukan, yang telah dikirimkan oleh *Route Agent*. Petunjuk orientasi ini bergantung pada lokasi pengguna dan perpindahannya.

7. KESIMPULAN

Pada sistem navigasi ruangan yang tradisional, pengguna masih diharuskan untuk menentukan posisi awal dan akhir dari rute navigasi secara tepat dan jelas. Berbeda dengan CAIN, pengguna bisa menemukan lokasi target yang belum diketahui sebelumnya untuk keadaan lingkungan yang bersifat dinamis. Penelitian ini mampu melakukan optimasi dalam hal jarak dan waktu pada tahapan pencarian jalur navigasi pada CAIN dengan menambahkan algoritma Koloni Semut untuk menemukan jalur navigasi yang terpendek.

Pada penelitian selanjutnya dapat dikembangkan beberapa algoritma pencarian jalur terpendek untuk memberikan hasil yang lebih baik pada sistem navigasi ruangan. Perbandingan performansi antara beberapa metode navigasi juga dapat dilakukan untuk mengevaluasi sistem navigasi yang telah ada. Skenario penelitian ini diharapkan dapat diujicobakan pada beberapa kondisi lingkungan gedung yang berbeda agar dapat dilakukan perbaikan pada kekurangan-kekurangan sistem ini

8. DAFTAR PUSTAKA

[1] Anagnostopoulos, C., Tsetsos, V., Kikiras, P., Hadjiefthymiades, S.P.: **OntoNav: (2005) A semantic indoor navigation system**. In: Proceeding of the first International Workshop on Managing Context Information in Mobile and Pervasive Environments.

[2] Liu, A.L., Hile, H., Kautz, H., Borriello, G., Brown, P.A., Harniss, M., Johnson, K.: (2006) **Indoor wayfinding: developing a functional interface for individuals with cognitive impairments**. In: Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility (Assets '06), Portland, Oregon, USA, ACM Press: 95 – 102.

[3] Lyardet, F., Grimmer, J., Muhlhauser, M.: (2006) **CoINS: Context sensitive indoor navigation system**. In: Proceeding of the Eight IEEE International Symposium on Multimedia (ISM'06).

[4] de Almeida, D.R., de Souza Baptista, C., de Andrade, F.G.: (2006) **Using ontologies incontext-aware applications**. In: Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA '06), IEEE Computer Society 349{353 Washington, DC, USA.

[5] Yuhana, U. L., Jih, W., Chang, H. W., Hsu, J. Y., Tsai, W. C.: (2010) **An Ontology-Based Model for Context-Aware Indoor Navigation**. The 6th International Conference on Information & Communication Technology and Systems.

[6] Ghoseiri, K., Nadjari, B.: (2010) **An ant colony optimization algorithm for the bi-objective shortest path problem**. At ScienceDirect : Applied Soft Computing 10 (2010) 1237-1246.

[7] Mocholi, J. A., Jaen, J., Catala, A., Navarro, E.: (2010) **An Emotionally Biased Ant Colony Algorithm for Pathfinding in Games**. At ScienceDirect :

Expert Systems with Applications 37 (2010) 4921-4927.

[8] Abowd, G.D., Atkeson, C.G., Hong, J., Long, S., Kooper, R., Pinkerton, M.: (1997) **Cyberguide: a mobile context-aware tour guide**. Wireless Networks 3(5) 421 – 433.

[9] Retscher, G., Thienelt, M.: (2004) **NAVIO - a navigation and guidance service for pedestrians**. Journal of Global Positioning Systems (CPGPS) 3(1-2) 208-217.

[10] Hsieh, W.T., Miller, N., Yang, Y.F., Chou, S.C., Steenkiste, P.: (2004) **Calculating walking distance in a hybrid space model**. In: First International Workshop on Advanced Context Modelling, Reasoning And Management.

[11] Want, R., Hopper, A., Falc~ao, V., Gibbons, J.: (1992) **The active badge location system**. ACM Transactions on Information Systems (TOIS) 10(1): 91-102.

[12] Schilit, B., Adams, N., Want, R.: (1994) **Context-aware computing applications**. In: Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, US: 85 – 90.

[13] Dey, A.K., Abowd, G.D., Salber, D.: (2001) **A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications**. Human-Computer Interaction 16: 97-166.

[14] Noy, N.F., McGuinnes, D.L.: (2001) **Ontology development 101 : A guide to creating your first ontology**. Technical report, Stanford University.

[15] You, C.w., Chen, Y.C., Chiang, J.R., Huang, P., Chu, H.h., Lau, S.Y.: (2006) **Sensorenhanced mobility prediction for energy-e_cient localization**. In: Proceedings of Third Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2006). Volume 2., Reston, VA, USA : 565-574.

[16] Russel, S.J., Norvig, P.: (2003) **Informed Search and Exploration**. In: Arti_cial Intelligence: A Modern Approach. Second edn. Prentice Hall : 94-136.

[17] Dorigo, M., Maniezzo, V., Colorni, A.: (1991) **Ant System: An Autocatalytic Optimizing Process**. Technical Report, Dipartimento di Elettronica e Informazione, Politecnico di Milano.

[18] Dorigo, M., Gambardella, L.M.: (1997) **A Cooperative Learning Approach to The Traveling Salesman Problem**. IEEE Transaction on Evolutionary Computation 1 53–66.

[19] Stützle, T., Dorigo, M.: (1999) **ACO Algorithms for The Traveling Salesman Problem.** Working Paper, Universite Libre de Bruxelles, Belgium,.

[20] Dorigo, M., Caro, G.D., Gambardella, L.M.: (1999) **Ant Algorithms for Discrete Optimization.** Artificial Life 5 (2) 137–172.

[21] Socha, T., Sampels, M., Manfrin, M.: (2003) **Ant Algorithms for The University Course Timetabling Problem with Regard to The State-of-The-Art.** In: Proceedings of the Third European Workshop on Evolutionary Computation in Combinatorial Optimization (EvoCOP), , pp. 334–345.