TEDC Vol.8 No.1 Januari 2014: 80-85

PENGGUNAAN METODE FORMAL PADA SISTEM FAULT TOLERAN INTEGRATED MODULAR AVIONICS

Ida Bagus Budiyanto
Jurusan Teknik Informatika Politeknik TEDC Bandung
E-mail: ibudiyanto@gmail.com

Abstrak
Tahap analisis dan perancangan merupakan tahap yang sangat penting dalam pengembangan sebuah sistem. Kesalahan pada tahap ini akan berdampak pada tahap berikutnya, dan pada akhirnya berdampak pada kegagalan sistem. Spesifikasi yang dibuat menggunakan bahasa alami memiliki banyak kelemahan seperti kontradiktif, rancu, samar, bermakna ganda, dan tidak lengkap. Metode formal yang menggunakan logika formal dan matematika mulai digunakan untuk mengatasi hal ini. Makalah ini menggambarkan beberapa pemakaian metode formal dalam sistem fault tolerance khususnya sistem avionik modular terintegrasi.

Kata kunci : metode formal, fault toleran, IMA.

Abstract
The analysis and design phaseis very important phase in the system development. The specifications which are made using natural language has many disadvantages, such as contradictory, vague, ambiguous, and incomplete. The weakness in this specification will continue to the next stage, and it will causethe system failure.Here is a formal method based on a mathematical model of the system is designed to be very helpful. Formal methods allow the designer to determine the specifications of the system at different abstraction levels and verify the consistency of this formal specification before it is implemented.This paper explores the using of formal method formal in the fault tolerance system, especially integrated modular avionics (IMA) system.

Key words: formal method, fault tolerance, IMA.

Pendahuluan

Pengembangan software untuk sistem waktu-nyata kritis dan aman (*safety-critical real-time system*) merupakan permasalahan yang sangat rumit.Kegagalan software pada sistem ini dapat membahayakan kehidupan manusia.Avionik, sebagai salah satu sistem waktu nyata kritis dan aman, merupakan sebuah sistem kompleks dalam pesawat terbang yang mengelola sistem navigasi, komunikasi, kendali terbang dan display data. Kegagalan fungsi pada sistem avionik ini akan berdampak kepada kegagalan terbang pada pesawat sehingga dapat menghilangkan nyawa penumpangnya.

Kegagalan fungsi pada sistem waktu-nyata dapat diminimalisir dengan mengembangkan sistem yang memiliki ketahanan terhadap kesalahan (*fault-tolerance*).Kesalahan yang terjadi dalam sistem dapat diprediksi dan diantisipasi sehingga tidak berlanjut menjadi kegagalan sistem.Fault-tolerance dapat dilakukan dengan redundansi hardware dan software. Redundansi hardware dapat dilakukan dengan menambah jumlah prosesor yang mengelola tugas yang sama, sedangkan redundansi software dapat dilakukan dengan membuat beberapa versi software yang melakukan tugas yang sama.

Model pengembangan perangkat lunak secara umum terdiri dari proses-proses: definisi masalah, analisis kebutuhan, desain, implementasi, integrasi dan pengujian. Tahap analisis kebutuhan merupakan tahap yang sangat penting, karena hasil dari tahapan ini yang berupa spesifikasi sistem akan digunakan sebagai dasar bagi tahap-tahap berikutnya. Spesifikasi yang dibuat dengan bahasa alami mempunyai banyak kelemahan, seperti kontradiktif, rancu, samar, bermakna ganda, dan tidak lengkap. Kelemahan pada spesifikasi ini akan berlanjut ke tahap berikutnya, sehingga dapat berakibat pada kegagalan sistem yang pada sistem kritis-aman dapat menghilangkan jiwa manusia.

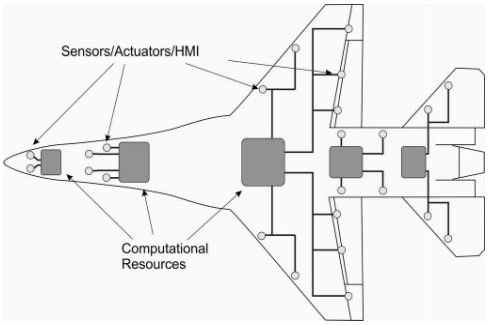
Pengembangan software untuk sistem waktu-nyata kritis yang kompleks menuntut ketelitian dari tahapan awal hingga akhir pengembangan.Kesalahan pada salah satu pentahapannya dapat berdampak fatal pada sistem akhirnya.Demikian juga pada pengembangan sistem avionik, yang saat ini berkembang ke arah modular dan terintegrasi (*integrated modular avionics*/IMA), diperlukan ketelitian pada semua tahap pengembangannya.

Sistem Avionik Pesawat Terbang

Sistem waktu nyata adalah sistem yang selalu berinteraksi dengan lingkungannya dan memberi respon dalam waktu tertentu. Pada umumnya sistem waktu nyata selalu aktif dan siap menerima input dari lingkungannya. Sedangkan sistem waktu nyata aman-kritis (*safety-critical real-time system*) adalah sistem waktu nyata yang apabila mengalami kegagalan fungsi dapat berdampak pada kehilangan nyawa manusia [1].

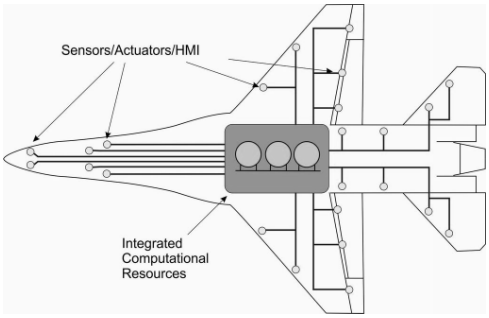
Sistem avionik klasik dikembangkan dengan memperhatikan ketelitian ini telah dilakukan oleh Bharadwaj dan Heitmeyer yang menggunakan metode kebutuhan SCR (*Software Cost Reduction*) yang merupakan salah satu metode formal yang berbasis pada tabel-tabel untuk spesifikasi dan analisis perilaku sistem yang diinginkan [2].

Pada konsep klasik, avionik dikembangkan secara tersebar (*federated avionics*) dimana masing-masing subsistem dilayani oleh komputer yang berbeda yang memang dikhususkan untuk subsistem tersebut. Setiap komputer pada *federated avionics* dapat memiliki spesifikasi perangkat keras yang berbeda, sehingga lebih menyulitkan saat pengujian dan perawatan, disamping banyaknya komputer ini tentu akan menambah beban pada pesawat terbang.



Gambar 1.*Federated Avionic*

Pada konsep baru, avionik berkembang ke arah sistem yang modular dan terintegrasi (*integrated modular avionics*/IMA) [3].IMA pertama kali dikenalkan oleh Honeywell yang dipasang pada pesawat Boeing 777 pada tahun 1995, memiliki keunggulan dibanding sistem avionik sebelumnya antara lain fleksibilitas, reusabilitas, modularitas dan biaya lebih murah [4].



Gambar 2.*Integrated Modular Avionic* (IMA)

Sistem komputer pada IMA menyediakan layanan sistem operasi seperti penjadwalan aplikasi-aplikasi yang berjalan pada sistem dan pengelolaan sumber daya yang dipakai bersama.Aplikasi-aplikasi ini tersebar ke dalam beberapa modul, namun secara fisik tersimpan dalam satu komputer.Sistem IMA menyediakan mekanisme untuk menjamin sumber daya dapat dibagi secara aman.

Analisis kesalahan pada IMA pernah dilakukan oleh Phillipa Conmy [5] yang yang menganalisis kesalahan pada tingkat tinggi, dimana kesalahan dibagi dalam tiga kategori, yaitu:

a.Perangkat I/O dan aplikasi sudah benar, tetapi berfungsi tidak benar

b.Respon dari sistem IMA kepada perangkat I/O dan aplikasi tidak benar

c.Kesalahan sistem IMA yang belum terprediksikan, seperti kesalahan pada IMA tanpa ada pemberitahuan dari aplikasi atau I/O.

Analisis dilakukan menggunakan kata petunjuk SHARD (*Software Hazard Analysis and Resolution in Design*) dan LISA (*Low-level Interaction Safety Analysis*).Analisis ini digunakan untuk menyediakan analisis keselamatan dari sistem komputer kritis-aman. SHARD menguji kemungkinan deviasi pada aliran informasi antar elemen software, sedangkan LISA menguji deviasi pada event dan penggunaan sumber daya sistem yang dibatasi waktu. Migrasi ke sistem avionik yang modular dan terintegrasi yang dilakukan oleh Audsley [6] menunjukkan adanya peningkatan fault-tolerance.

Permasalahan yang muncul pada sistem avionik modular dan terintegrasi diantaranya adalah masalah alokasi sumber daya, karena dalam IMA sebuah sumber daya akan digunakan secara bersama-sama oleh beberapa modul aplikasi. Salah satu pendekatan untuk mengatasi masalah alokasi sumber daya ini dilakukan dengan alokasi sumber daya secara hirarkis yang disampaikan oleh T. Zhou et. al. [7].Pertama, waktu respon terburuk dari task-task dengan deadline sembarang dianalisis untuk penjadwal dua tingkat.Kemudian, pendekatan alokasi sumber daya hirarkis disajikan dalam dua tingkat. Pada tingkat platform, algoritma penentuan task yang memperoleh sumber daya dilakukan berdasarkan *genetic simulated annealing* (GSA) diusulkan untuk menetapkan satu set task yang ditentukan untuk node pengolahan yang berbeda, sehingga sumber daya dapat dialokasikan sebagai partisi dan dipetakan ke kelompok task. Dengan keterbatasan sumber daya, algoritma mencoba untuk menemukan task yang optimal dengan biaya komunikasi minimal dan beban kerja yang seimbang.Pada tingkat node, parameter partisi yang dioptimalkan, sehingga sumber daya komputasi dapat dialokasikan lebih lanjut.

Fault Tolerance

Sistem IMA melayani sejumlah komputasi aplikasi yang saling berkomunikasi pada jaringan bersama.Aplikasi yang berjalan bersama ini harus berbagi sumberdaya seperti processor dan memory, sehingga diperlukan mekanisme untuk menjamin hal ini dapat terjadi dengan baik.Sistem Fault tolerance tetap berjalan walapun terjadi kegagalan pada beberapa bagian sistem, sehingga harus memiliki sumber daya cadangan untuk menjalankan sistem. Terdapat dua cara bagi sistem untuk tahan terhadap kegagalan, yaitu [8]

a.Hardware: teknik ini mengandalkan pada penambahan hardware redundan pada sistem.

b.Software: teknik mengandalkan pada penggandaan kode program atau proses.

Fault Tolerance Hardware

Sebagian besar desain fault-tolerant dilakukan dengan mengembangkan komputer yang yang secara otomatis dapat melakukan perbaikan (recover) jika terjadi kesalahan secara acak pada komponen hardware.Teknik yang digunakan diantaranya adalah partisioning sistem komputasi ke dalam modul-modul yang berperan sebagai daerah pengendali kesalahan.Setiap modul memiliki cadangan (redundancy), sehingga jika terjadi kegagalan modul, maka dapat digantikan oleh cadangannya.Untuk keperluan ini, ditambahkan pula sebuah mekanisme khusus yang dapat mendeteksi kesalahan dan melakukan recovery.Pendekatan yang sering digunakan adalah fault-masking dan dynamic recovery.

Fault masking adalah teknik redundansi dimana sejumlah modul identik menjalankan fungsi yang sama, dan outputnya dipilih untuk menggantikan modul yang salah. *Triple modular redundancy* (TMR) sering digunakan untuk melakukan fault-masking, dimana suatu fungsi dikerjakan oleh tiga modul, kemudian dipilih salah satunya. Sistem TMR dianggap gagal jika dua diantara tiga modul redundan terjadi kesalahan, sehingga hasil pemilihan menjadi tidak valid.

Dynamic recovery memiliki suatu mekanisme untuk mendeteksi kesalahan yang mungkin terjadi pada modul-modul, kemudian mengalihkan fungsinya pada modul cadangan yang akan melakukan aksi (*rollback, initialization, retry, restart*) yang diperlukan untuk melanjutkan komputasi. *Dynamic recovery* biasanya lebih efisien dibandingkan fault-masking, karena itu pendekatan ini sering dipilih terutama pada sistem dengan sumber daya terbatas. Tetapi memiliki kelemahan adanya penundaan waktu komputasi yang terjadi selama proses *recovery* kesalahan.

Fault-Tolerance Software

Fault-tolerance pada software dibuat dengan menggunakan pendekatan redundansi statis dan dinamis seperti yang digunakan pada hardware. Salah satu pendekatan adalah pemrograman N-version yang menggunakan redundansi statis, dimana program ditulis dalam beberapa versi yang saling independen yang melakukan fungsi yang sama, kemudian dipilih outputnya. Tentu saja output dari beberapa modul yang dipilih tidak sama persis, sehingga harus digunakan kriteria untuk mengidentifikasi dan menolak versi yang salah dan menentukan sebuah nilai yang konsisten yang dihasilkan olehmodul-modul yang baik.

Alternatif pendekatan dinamis dilakukan berdasarkan konsep blok pemulihan dimana program dibagi ke dalam blok-blok, kemudian dilakukan pengujian pada semua blok. Jika hasil pengujian gagal, blok kode redundan akan dijalankan. Pendekatan lain dilakukan dengan mengombinasikan fault-tolerance hardware dan software dengan mengimplementasikan sistem komputer fault tolerance menggunakan hardware dan software yang berbeda, serta sebuah metode untuk mengidentifikasi sistem yang deviasinya tidak diterima oleh yang lain. Ini merupakan teknik yang cukup mahal, namun sering digunakan pada aplikasi yangsangat kritis seperti pada sistem kendali pesawat terbang.

Beberapa Algoritma Fault-Tolerance

Ghosh et. al. [9] menawarkan algoritma penjadwalan fault-tolerance pada sistem multiprosesor yang menjamin kelengkapan penjadwalan task-task sebelum terjadi deadline. Algoritma menjadwalkan beberapa task cadangan yang saling overlap dan secara dinamis mendealokasi task cadangan ketika task aslinya telah selesai dieksekusi, ini akan menambah utilisasi prosesor.

Rubel et.al.mengembangkan teknologi fault tolerance untuk sistem DRE (*distributed real-time embedded*), yang memberikan tiga teknik fault tolerance yang mendukung kebutuhan sistem DRE, yaitu pendekatan transparan pada komunikasi mode campuran, auto-konfigurasi untuk sistem dinamik, dan duplikasi menejemen interaksi peer-to-peer. Kemudian dijelaskan kemampuan fault-tolerance terintegrasi untuk sistem DRE berbasis komponen dunia nyata yang menggunakan middleware yang tersedia di pasar.Hasil eksperimen menunjukkan bahwa kemampuan sistem yang dibuat memenuhi kebutuhan kinerja waktu nyata sistem DRE untuk tanggapan terhadap recovery kegagalan [10].

Pendekatan lain dilakukan oleh Modarressi et.al. untuk*appication-specific instruction-set processor* (ASIP) yang bertujuan mengurangi biaya pada mekanisme fault-tolerant klasik. Pendekatan fault-tolerance yang diusulkan, mendeteksi kesalahan fungsional pada sebuah unit hardware, unit ini diganti dengan versi software yang memiliki fungsi yang sama. Untuk mengevaluasi pendekatan fault-tolerance, digunakan prosesor JPEG [11].Hasil analitis dan eksperimen menunjukkan bahwa pendekatan ini dapat mengatasi kesalahan unit fungsional tanpa tambahan biaya dan tempat.

Fault-tolerance pada hardware dapat dilakukan menggunakan multipel prosesor untuk mendapatkan kinerja tinggi dan unit memory dual port yang digunakan untuk komunikasi antar-prosesor. Khan mengusulkan sistem HPEC (*high perfarmance embedded computer*) yang terdiri dari lima prosesor yang dipartisi menjadi dua kelompok, yaitu partisi komputasi dan partisi IO [12]. Partisi komputasi fokus pada task intensif komputasional, terdiri dari tiga prosesor pekerja.Sedangkan, partisi IO melaksanakan tugas umum dan task yang berhubungan dengan IO real-time. Partisi IO ini memiliki dua prosesor interface dengan kemampuan IO kecepatan tinggi dan proses interupsi yang cepat. Prosesor inti untuk partisi-partisi ini dipilih berdasarkan fungsional komputasi dan IO kecepatan tinggi. Arsitektur HPEC ini fault tolerant dalam melakukan pengendalian terhadap kesalahan dan isolasi terhadap unit yang bermasalah.

Pendekatan lain ditawarkan oleh Salim Kalla yang mengambil input sekelompok task, target berarsitektur terdistribusi, beberapa batasan distribusi, beberapa indikasi pada waktu eksekusi operasi-operasi pada prosesor target, beberapa indikasi pada waktu komunikasi pada target, dan realibilitas prosesor [13].Arsitektur ini menghasilkan fault-tolerant terdistribusi dan penjadwalan statis, dengan indikasi apakah batasan real-time terpenuhi atau tidak. Pendekatan penjadwalan yang diusulkan terkait dengan channel yang berisi daftar penjadwalan heuristik berdasarkan strategi replikasi aktif berbasis *Global System Failure Rate* (GSFR). GSFR adalah rata-rata kesalahan per satuan waktu dari penjadwalan prosesor.GSFR ini sangat baik pada penjadwalan yang dieksekusi secara periodik.

Pendekatan fault tolerance terhadap software dapat dialikasikan pada beberapa lapisan software, seperti pada level sistem operasi, level fungsi/method, level proses, atau level objek. Pendekatan yang ditawarkan oleh Afonso yang membangun framework berdasarkan layanan sistem operasi dan komunikasi middleware [14]. Framework fault tolerant yang dibangun diintegrasikan dengan sistem operasi embedded real-time (BOSS), berada pada level thread.

Metode Formal pada Pesawat Terbang

Pemodelan rancangan sistem IMA dilakukan oleh [15] menggunakan *Generic Model Environment* (GME) yang merupakan perangkat berorientasi objek yang dapat dikonfigurasi untuk pemodelan domain khusus dan lingkungan sintesa program.Pendekatan ini disebut MIMAD (*Modeling paradigm for Integrated Modular Avionics Design*). Konsep metamodeling

diusulkan dalam GME untuk menggambarkan paradigma pemodelan untuk domain tertentu, termasuk konsep dasar yang diperlukan untuk mewakili model dari kedua sudut pandang sintaksis dan semantical.

Metode formal juga pernah digunakan pada sistem dalam pesawat, seperti pada sistem kendali terbang yang dikembangkan oleh [16] menggunakan Simulink dan diverifikasi menggunakan pemeriksa model NuSMV. SCADE Suite dan Reactis digunakan untuk menggabungkan software yang dikembangkan oleh RCI dan Universitas Minnesota untuk menerjemahkan model simulink ke NuSMV. Dari penelitian yang dilakukan, diagram transisi mode yang dibuat dengan Simulink dapat secara otomatis diterjemahkan ke NuSMV dan dianalisa oleh pemeriksa model NuSMV.

Verifikasi formal untuk sistem kendali terbang juga dipakai oleh [17] yang menggunakan metode ICA (*Interactive Convergence Algorithm*) untuk melakukan verifikasi formal terhadap algoritma yang mendukung fault-tolerance pada sistem flight control digital. Metode lain untuk sistem kendali terbang digunakan oleh [18] yang menggunakan metode B.

Spesifikasi Formal

Spesifikasi adalah proses penggambaran sebuah sistem beserta sifat-sifat yang diinginkan. Sifat-sidat sistem meliputi perilaku fungsional, perilaku pewaktuan, karakteristik kinerja atau struktur internal sistem tersebut.Sedangkan spesifikasi formal adalah spesifikasi yang menggunakan bahasa yang diungkapkan dengan kalimat matematika.

Beberapa jenis metode formal seperti notasi Z, VDM dan Larch memfokuskan pada perilaku sistem sekuensial. State digambarkan dalam bentuk struktur matematika seperti himpunan, relasi dan fungsi. Transisi state digambarkan dengan pre dan post-condition. Metode lain seperti CSP, CCS, Statecharts, Temporal Logic, dan I/O Automata fokus pada perilaku sistem konkuren, dimana state dinyatakan dalam domain sederhana seperti bilangan integer atau dibiarkan tidak terintepretasi. Metode lain seperti RAISE dan LOTOS. Spesifikasi formal dapat juga menggunakan Real-Time Logic (RTL)

Verifikasi Formal

Pendekatan verifikasi formal yang sudah baku adalah model checking dan theorem proving, dimana metode formal digunakan untuk menganalisis sebuah sistem untuk sifat yang diinginkan.

Model checking merupakan teknik yang mengacu pada finite model sebuah sistem dan memeriksa bahwa sifat yang diinginkan sudah dimiliki oleh model.Pendekatan model checking yang banyak dipakai adalah pertama temporal model checking¸ dimana spesifikasi diekspresikan dalam logika temporal dan sistem dimodelkan sebagai sistem finite state transition.Pendekatan kedua, spesifikasi dinyatakan sebagai sebuah automata, dibandingkan dengan spesifikasi untuk menentukan apakah perilaku memenuhi spesifikasi yang diharapkan. Kelemahan model checking adalah masalah state explosion, seperti yang pernah terjadi pada McMillan yang menggunakan binary decision diagrams (BBD) untuk merepresentasikan transisi state dari sistem, dimana peningkatan ukuran sistem menjadi tidak terverifikasi.

Untuk memenuhi teknik model checking pada sistem terbuka, diperlukan pemodelan lingkungan yang sesuai dengan alam nyata, misalnya dari sisi kendala waktu.Asumsi terhadap lingkungan dibatasi oleh perilaku dari modul [19].Pendekatan perilaku dapat secara efisien meningkatkan kebenaran dan reliabilitas sistem.Behavior Protocol (BP) digunakan dalam model komponen SOFA dengan menyertakan informasi batasan waktu [20].

Theorem Proving adalah teknik dimana sistem dan sifatnya diekspresikan sebagai formula dalam logika matematika. Logika ini diberikan oleh sistem formal yang menetukan himpunan aksioma dan aturan.Theorem proving merupakan proses menemukan pembuktian sifat-sifat dari aksioma-aksioma sistem.

Pengembangan sistem fault-tolerant terdistribusi menggunakan pendekatan formal diantaranya menggunakan formalisme sistem aksi sebagai framework perancangan [21].Formalisme menyediakan framework untuk mengembangkan sistem terdistribusi reaktif yang menggunakan teknik perbaikan bertahap (stepwise refinement).Selama pengembangan sistem dengan teknik perbaikan bertahap, dimulai dengan spesifikasi abstrak dan merincinya ke dalam program yang dapat dieksekusi dalam sejumlah langkah yang benar.Teknik perbaikan bertahap mengijinkan untuk memasukkan kebutuhan sistem ke dalam spesifikasi secara bertahap dan hingga pasa saat implementasi sistem. Langkah-langkah perbaikan secara bertahap mengenalkan representasi detil dari mekanisme fault tolerance ke dalam spesifikasi abstrak dari komponen dan sistem keseluruhan.

Kesimpulan

Metode formal menjadi alternatif yang baik untuk pengembangan sistem yang dapat mengatasi kelemahan-kelemahan penggunaan bahasa alami.Saat ini berkembang beberapa jenis metode formal baik untuk spesifikasi formal dan verifikasi formal, yang dikembangkan oleh personal atau kelompok/institusi sesuai dengan kebutuhan masing-masing. Namun, metode formal berpotensi menemukan kendala saat akan diterjemahkan ke dalam implementasi pemrograman, terutama apabila tim pemrogramnya tidak memahami logika formal dan matematika dengan baik.

Daftar Pustaka

[1]. E. R. Olderog, Real-Time Systems: Formal Specification and Automatic Verification. Cambridge University Press, 2008, p. 338.

[2].R.Bharadwaj and C. Heitmeyer, "DEVELOPING HIGH ASSURANCE AVIONICS SYSTEMS WITH THE SCR REQUIREMENTS METHOD*," Office, no. October, pp. 7-13, 2000.

[3].H. Butz, "Open Integrated Modular Avionic (IMA): State of the Art and future Development Road Map at Airbus Deutschland Function SW in MB Signal Interfaces x 1000," Processing, 2006.

[4]. F. Belli, M. E. Innocenti, and L. S. E. Vladova, "STEP-BY-STEP MIGRATION FROM FEDERATED AVIONICS TO IMA SYSTEM," Development, pp. 3-4, 2009.

[5]. P. Conmy and J. Mcdermid, "High level failure analysis for Integrated Modular Avionics," Australian Computer Society, 2001.

[6]. N. C. Audsley and M. Burke, "Distributed Fault-Tolerant Avionic Systems – A Real-Time Perspective," Computing Systems, 1997.

[7]. T. Zhou, H. Xiong, and Z. Zhang, "Hierarchical Resource Allocation for Integrated Modular Avionics Systems," Journal of Systems Engineering and Electronics, vol. 22, no. 5, pp. 780-787, 2011.

[8]. A. C. Persya and T. R. G. Nair, "Fault Tolerant Real Time Systems," International Conference, p. 4, 2008.

[9]. S. Ghosh, R. Melhem, and D. Mosse, "Fault-Tolerant Scheduling on a Hard Real-Time Mutiprocessor System," International Parallel Processing Symposium, 1994.

[10]. P. Rubel, M. Gillen, R. Schantz, and A. Paulos, "Fault-Tolerant Approaches for Distributed Real-Time and Embedded System," DARPA, 2006.

[11]. M. Modarressi et al., "A Fault-Tolerant Approach to Embedded-System Design Using Software Standby Sparing," Computer, 2005.

[12]. G. N. Khan and N. Avenue, "Fault-Tolerant Architecture for High Performance Embedded System Applications," Conference on Computer Design, vol. 5, pp. 384-389, 1998.

[13]. S. Kalla, "Reliability-Driven Fault Tolerant Scheduling Heuristics for Distributed Embedded Real-Time Systems," International Journal of Coputer Applications, vol. 36, no. 5, pp. 5-11, 2011.

[14]. F. Afonso, "Application-Level Fault Tolerance in Real-Time Embedded Systems," Industrial Electronics, 2007.

[15]. A. Gamati, B. Romain, D. Thierry, G. J.-pierre Talpin, I. Futurs, and C. Lezennes, "A Modeling Paradigm for Integrated Modular Avionics Design," Network, 2006.

[16]. S. P. Miller, E. A. Anderson, L. G. Wagner, and M. W. Whalen, "Formal Verification of Flight Critical Software," Lustre, pp. 1-16, 2005.

[17]. J. Rushby and F. V. Henke, "Formal Verification of Algorithms for Critical Systems ∗ Flight Control Systems," Engineering, vol. 19, no. 5, pp. 1-15, 1993.

[18]. A. M. Wahba, "Formal Verification of Real-Time Distributed Systems Using B Method," International Journal of Engineering Science, vol. 3, no. 4, pp. 3427-3436, 2011.

[19]. A. Banerjee, P. Dasgupta, and P. P. Chakrabarti, "Formal verification of modules under real time environment constraints," IEEE Computer Society, pp. 103-108, 2004.

[20]. Y. Jia, Z. Zhang, and S. Xie, "Formal Specification and Verification of Components Real-time Behavior," IEEE, no. 90718017, pp. 198-201, 2010.

[21]. E. Troubitsyna, "Formal Specification of Fault Tolerant Distributed Systems in the Action Systems Formalism," International Conference on Communication Theory, Reliability and Quality of Service, pp. 139-143, 2010.