Menganalisis Kesalahan Kebutuhan Perangkat Lunak dalam Sistem Tertanam, Kritis Keamanan

## Robyn R. Lutz '

Laboratorium Propulsi Jet Institut Teknologi California Pasadena, CA 91109

**Abstrak**

#### potensi bahaya aafety (13]. Untuk penelitian ini 8 kesalahan perangkat lunak pada Voyager dan 122 kesalahan perangkat lunak pada

*Kertas tipis anolyzee akar cauot8* **0 /** 40 / cry-relatedGalileo didokumentasikan dalam **memiliki** berpotensi signifikan

soJu ›adalah kesalahan dalam anJefy-criiict @ , gM§gdd6d 4**8itM4. DARI** Efek CataatrOphic diklasifikasikan sebagai terkait dengan keselamatan.

***Reoulto zltow itu*** *oo kesalahan perangkat diidentifikasi ao*

sistem ieffy haza rdou z ke lhc yang kuat dipinjamkan menjadi Rro-

*duced oleh *



*kesalahan perangkat lunak terkait. Kesalahan eoflware terkait la Jely terbukti paling kriminal*umumnya / rotn (1) perbedaan percaya pada dokumen Yequiremc •! \* \* p \*\*! 'L-



*bersandar dari lhe* **gysfem dan** *(fl) kesalahpahaman tentang lhe so / ft // ore's $ nlerface willt lheel of lhe syslem. Itu***penggunaan kertas** fkeye rest ‹Its la idenfi / y terpenuhi / tods Lg trhich persyaratan emors ca ll \* • Dipersiapkan. Tujuan Z'he adalah la

*kurangi kesalahan soflware yang berhubungan dengan aman «***ri *d*** *la enhvnc ‹*

*Dia aman dari • Permohonan, sistem tertanam.*

# pengantar

**Pesawat ruang angkasa** perangkat lunak sangat penting bagi keselamatan karena ia memantau dan mengontrol komponen yang dapat terlibat dalam perilaku sistem yang berbahaya (11]. Perangkat lunak harus dijalankan dalam konteks sistem tanpa menimbulkan risiko yang tidak dapat diterima.

Setiap pesawat ruang angkasa melibatkan peranti lunak tertanam yang didistribusikan pada beberapa komputer penerbangan yang berbeda. Voyages memiliki sekitar 18.000 baris kode sumber; Galileo memiliki lebih dari 22.000 (18]. Perangkat lunak yang tertanam adalah perangkat lunak yang dijalankan pada komputer ayBtem yang merupakan bagian integral dari sistem yang lebih besar yang tujuan utamanya bukan untuk komputasi [6]. Perangkat lunak pada kedua pesawat ruang angkasa ini sangat interaktif dalam hal derajat penyampaian pesan antar komponen sistem, kebutuhan untuk merespon secara real-time

Makalah ini membahas 387 kesalahan perangkat lunak terbongkar untuk memantau perangkat keras dan lingkungan, dan

selama integrasi dan pengujian sistem dua ruang- masalah waktu yang kompleks di antara bagian-bagian sistem.

kerajinan, Voyager dan Galileo. Kesalahan perangkat lunakPengembangan perangkat lunak tim untuk setiap pesawat ruang angkasa untuk menjadi perbedaan terkait perangkat lunak antara biaya yang melibatkan banyak tim yang bekerja selama beberapa tahun. nilai atau kondisi yang diletakkan, diamati, atau diukur dan

nilai yang benar, ditentukan, atau secara teoritis benar atau

kondisi (6]. Masing-masing kesalahan perangkat lunak ini docu-Tujuan dari makalah ini adalah untuk mengidentifikasi keterangan pada saat ditemukan oleh a bentuk menggambarkan tenda dan cara-cara di mana sebab / akibat berhubungan- masalah atau kegagalan. Formulir juga mencatat kesalahan perangkat lunak terkait keselamatan Kapal berbeda dari

analisis selanjutnya dan perbaikannya tindakan yang diambil. hubungan sebab / akibat dari perangkat lunak yang tidak terkait dengan keselamatan- Sebagai bagian dari prosedur standar untuk memperbaiki kesalahan perangkat lunak. Secara khusus, analisis menunjukkan hal itu er-

setiap kesalahan perangkat lunak yang dilaporkan, efek kegagalan setiap kesalahan dalam mengidentifikasi atau memahami fungsional dan in- diklasifikasikan sebagai dapat diabaikan, signifikan, atau bencana. persyaratan terface sering mengarah ke yang berhubungan dengan keselamatan. Yang diklasifikasikan sebagai signifikan atau katastropik ada di Kesalahan perangkat lunak. Perbedaan ini digunakan untuk mengidentifikasi sisa-sisa oleh analis keamanan sistem sebagaimewakili metode dimana penyebab umum terkait keselamatan

kesalahan perangkat lunak dapat menjadi sasaran selama pengembangan.

\* Alamat surat penulis adalah Dept. of Computer Science, Iowa State University, Ames, IA 5 £ D11. Pencarian yang dijelaskan dalam makalah ini dilakukan oleh Jet Propulsion Laboratory,

Tujuannya adalah untuk meningkatkan keamanan sistem dengan memahami dan, jika memungkinkan, menghilangkan sumber yang lazim

Institut Teknologi California, di bawah kontrak dengan NASA. Perangkat lunak yang berhubungan dengan keselamatan kesalahan.

126

0-8186-3120-1 / 92 $ 3. iXi O 1992 IEEE

### Metodologi

Studi yang dijelaskan di sini mencirikan akar penyebab kesalahan perangkat lunak terkait keselamatan yang ditemukan selama integrasi dan pengujian sistem. Pekerjaan terbaru oleh N akajo dan Kume pada hubungan sebab / akibat kesalahan perangkat lunak menawarkan kerangka kerja yang sesuai untuk mengklasifikasikan kesalahan perangkat lunak (16]. Pekerjaan mereka diperluas di sini untuk menjelaskan kompleksitas tambahan yang beroperasi secara luas, , sistem yang tertanam dengan persyaratan yang terus berkembang didorong oleh masalah perangkat keras dan lingkungan.

Skema klasifikasi Nakajo dan Kume menganalisis tiga titik jalur dari belakang kesalahan perangkat lunak ke sumbernya. Pendekatan ini memungkinkan klasifikasi tidak hanya kesalahan perangkat lunak yang didokumentasikan (disebut kesalahan program), tetapi juga kesalahan manusia sebelumnya (akar penyebab, misalnya, kesalahpahaman spesifikasi antarmuka), dan, sebelum itu , dari kekurangan proses yang berkontribusi pada kemungkinan terjadinya kesalahan (misalnya, komunikasi yang tidak memadai antara tim rekayasa sistem dan pengembangan perangkat lunak). Skema klaesifikasi dengan demikian mengarah ke belakang dalam waktu dari kesalahan perangkat lunak yang jelas ke analisis akar penyebab (biasanya kesalahan komunikasi atau kesalahan dalam mengenali atau menerapkan persyaratan), ke analisis proses pengembangan perangkat lunak. Dengan membandingkan mekanisme kesalahan umum untuk kesalahan perangkat lunak yang diidentifikasi berpotensi berbahaya dengan kesalahan perangkat lunak lainnya, akar penyebab umum dari kesalahan terkait keselamatan diisolasi. Klasifikasi sumber kesalahan kemudian diterapkan di sini untuk menentukan tindakan pencegahan yang dapat mencegah terjadinya kesalahan serupa dalam sistem tertanam yang kritis terhadap keselamatan. Makalah ini kemudian menggunakan skema klasifikasi untuk mengumpulkan profil kesalahan dari kesalahan perangkat lunak terkait keselamatan dan untuk mengidentifikasi metode pengembangan sistem tanam. Makalah ini kemudian menggunakan skema klasifikasi untuk mengumpulkan profil kesalahan dari kesalahan perangkat lunak terkait keselamatan dan untuk mengidentifikasi metode pengembangan sistem tertanam. Makalah ini kemudian menggunakan skema klasifikasi untuk mengumpulkan profil kesalahan dari kesalahan perangkat lunak yang berhubungan dengan keselamatan dan untuk mengidentifikasi metode pengembangan

dimana sumber kesalahan ini dapat dikontrol.

# Analisis Kesalahan Perangkat Lunak Terkait Keselamatan

### A. Tinjauan Skema Klasifikasi

Berikut gambaran skema klasifikasi, disesuaikan dengan kebutuhan perangkat lunak tertanam yang sangat penting bagi keselamatan. Lihat (16] untuk rincian tambahan tentang bagaimana kesalahan dikategorikan. Penyelidikan multi-proyek yang sedang berlangsung akan membahas masalah pengulangan (apakah analis yang berbeda mengklasifikasikan kesalahan yang diberikan dengan cara yang sama?).

* Kesalahan Program (Kesalahan Perangkat Lunak yang Didokumentasikan)

1. Kesalahan Internal (misalnya, sintaks)
2. Kesalahan Antarmuka (interaksi dengan komponen sistem lain, seperti transfer data atau

COfltFO)

1. Kesalahan Fungsional (kesalahan operasi: kelalaian atau operasi yang tidak perlu; kesalahan bersyarat: kondisi tidak benar atau nilai batas; kesalahan perilaku: perilaku tidak benar, tidak sesuai dengan persyaratan)

* Kesalahan Manusia (Akar Penyebab)

A. Kesalahan Pengkodean atau Pengeditan

#### B1. Kesalahan Komunikasi dalam Tim (salah memahami spesifikasi antarmuka S / W)

B2. Kesalahan Komunikasi Antar Tim(kesalahpahaman spesifikasi antarmuka H / W- tions atau spesifikasi S / W tim lain)

C1. Kesalahan dalam Mengenali Persyaratan (salah memahami spesifikasi atau masalah do- mâz)

C2. Kesalahan dalam Menerapkan Persyaratan (masalah menerapkan atau menerjemahkan persyaratan ke dalam desain)

* Cacat Proses (Cacat dalam Pengendalian Komando Sistem plexity + Ketidakcukupan dalam Komunikasi atau Metode Pengembangan)

1. Metode Pemeriksaan dan Pengujian Kode yang Tidak Memadai

DUA. Spesifikasi Antarmuka Tidak Memadai + Tidak Adilmenyamakan Komunikasi (di antara pengembang S / W opers)

B2. Spesifikasi Antarmuka Tidak Memadai -i- Komunikasi yang Tidak Memadai (antara pengembang S / W dan H / W)

Cl. Persyaratan Tidak Diidentifikasi atau Dipahami

+ Dokumentasi Tidak Lengkap

C2. Persyaratan Tidak Diidentifikasi atau Dipahami

+ Desain yang Tidak Memadai

Jelas, atribusi dari kesalahan manusia utama dan cacat proses utama untuk setiap kesalahan perangkat lunak yang terlalu menyederhanakan fies hubungan sebab / akibat. Namun, identifikasi faktor-faktor ini memungkinkan adanya karakterisasikesalahan perangkat lunak terkait keselamatan dengan cara yang menghubungkan fitur proses pengembangan dan sistem yang sedang dikembangkan dengan konsekuensi keselamatan fitur tersebut. Demikian pula asosiasi setiap kesalahan perangkat lunak dengan kesalahan manusia, sementara tidak realistis (dalam format

Apa arti kegagalan untuk memprediksi rincian perilaku sistem atau kesalahan?), memungkinkan hubungan yang berguna antara faktor manusia (seperti kesalahpahaman persyaratan atau realitas fisik yang mendasarinya) dan konsekuensi terkait keselamatan mereka.

1. **Kesalahan Program**

Kesalahan perangkat lunak yang terkait dengan keselamatan mencapai 56% dari total kesalahan perangkat lunak untuk Voyager dan 48% dari total kesalahan perangkat lunak untuk Galileo yang ditemukan selama integrasi dan pengujian sistem. (Pembaca mengacu pada [14] untuk tabel yang berisi data kesalahan.)

Beberapa kesalahan internal (mis., Kesalahan pengkodean internal ke modul perangkat lunak) ditemukan selama integrasi dan pengujian sistem. Pemeriksaan kesalahan perangkat lunak yang ditemukan kemudian selama operasi juga menunjukkan beberapa kesalahan internal. Tampaknya ertor pengkodean ini terdeteksi dan diperbaiki sebelum pengujian sistem dimulai. Dengan demikian, mereka tidak dibahas lebih lanjut dalam makalah ini.

Pada tingkat detail yang tinggi, kesalahan perangkat lunak yang berhubungan dengan keselamatan dan yang tidak terkait dengan keselamatan menampilkan proporsi yang sama dari antarmuka dan kesalahan fungsional. Fun ction al fanffs (perbedaan operasi, bersyarat, atau perilaku dengan persyaratan fungsional) adalah jenis kesalahan perangkat lunak yang paling umum. Kesalahan perilaku terhitung sekitar setengah dari semua kesalahan fungsional pada kedua pesawat ruang angkasa (52% di Voyager; 47% di Galileo).

Di Voyager, separuh dari kesalahan fungsi yang terkait dengan keselamatan disebabkan oleh kesalahan perilaku (perangkat lunak berperilaku tidak benar). Di Galileo, persentase yang sedikit lebih besar disebabkan oleh kesalahan operasi (hampir selalu merupakan operasi yang diperlukan tetapi dihilangkan dalam peranti lunak) daripada kesalahan perilaku. Seringkali operasi yang dihilangkan melibatkan kegagalan untuk melakukan pemeriksaan kewajaran yang memadai pada input data ke modul. Hal ini sering mengakibatkan rutinitas pemulihan kesalahan dipanggil secara tidak tepat.

Kesalahan bersyarat (hampir selalu nilai yang salah pada suatu kondisi atau batas) cenderung terkait dengan keselamatan pada kedua pesawat ruang angkasa (total 73%). Nilai yang salah (mis., Deadbands atau delay timer) sering kali menimbulkan risiko bagi pesawat ruang angkasa karena memicu respons pemulihan kesalahan yang tidak tepat atau dengan gagal memicu respons yang dibutuhkan. Hubungan antara kesalahan bersyarat dan kesalahan perangkat lunak yang terkait dengan keselamatan menekankan pentingnya menentukan nilai yang benar untuk data apa pun yang digunakan dalam keputusan pengendalian dalam perangkat lunak yang melekat pada keselamatan dan kritis.

Analisis juga mengidentifikasi kesalahan inl e mace (interaksi yang tidak tepat dengan komponen sistem lain, seperti waktu atau transfer data atau kontrol) sebagai masalah yang signifikan (36% dari program terkait keselamatan

kesalahan pada Voyager; 19% di Galileo). Sekte. IV di bawah ini menjelaskan bagaimana insiden tinggi kesalahan antarmuka dalam sistem tertanam yang kompleks ini kontras dengan insiden rendah kesalahan antarmuka dalam studi sebelumnya pada perangkat lunak mandiri yang lebih sederhana.

1. **Hubungan Antara Kesalahan Program dan Akar Penyebab**

Langkah kedua dalam analisis sebab / akibat adalah menelusuri ke belakang dalam waktu ke faktor manusia yang terlibat dalam kesalahan program yang ditemukan selama integrasi dan pengujian sistem.

Untuk kesalahan antarmuka, faktor utama manusia adalah kesalahan komunikasi dalam tim pengembangan atau kesalahan komunikasi antara tim pengembangan dan tim lain. Dalam kasus terakhir, perbedaan lebih lanjut dibuat antara kesalahpahaman dan spesifikasi antarmuka perangkat keras dan kesalahpahaman, dan spesifikasi antarmuka dari imponen perangkat lunak lain. Kesalahan komunikasi antara tim pengembangan (bukan dalam tim) adalah penyebab utama kesalahan antarmuka (93% di Voyager, T2% di G xlileo). Kesalahan antarmuka terkait keselamatan dikaitkan atau sangat terkait dengan kesalahan komunikasi antara tim pengembangan dan lainnya (seringkali antara pengembang perangkat lunak dan insinyur sistem), bukan dengan kesalahan komunikasi dalam tim.

Perbedaan yang signifikan dalam distribusi penyebab kesalahan antara kesalahan antarmuka yang berhubungan dengan keselamatan dan yang tidak terkait dengan keselamatan muncul. Penyebab utama dari kesalahan aturan yang terkait dengan keselamatan adalah spesifikasi antar jalur yang tidak jelas (67% pada Voyager; 48% pada

G alileo). Contohnya adalah kesalahan yang disebabkan oleh kesalahan sebagai-

asumsi tentang keadaan awal relai atau dengan pola waktu detak jantung yang tidak diharapkan dalam mode operasi tertentu. (Di sisi lain, akar penyebab kesalahan antarmuka yang tidak terkait dengan keselamatan didistribusikan secara lebih merata antara spesifikasi perangkat keras yang disalahpahami dan spesifikasi perangkat lunak yang disalahpahami.) Profil kesalahan antarmuka terkait keselamatan yang dikumpulkan di sini menekankan pentingnya memahami perangkat lunak sebagai satu set komponen tertanam dalam sistem yang lebih besar.

Penyebab utama dari kesalahan klien yang berhubungan dengan keselamatan adalah kesalahan dalam mengingat persyaratan yang ada (62% di Voyager, 79% di Galileo). Kesalahan kondisional terkait keselamatan, misalnya, hampir selalu disebabkan oleh kesalahan dalam mengenali persyaratan. (Di sisi lain, kesalahan fungsional yang tidak terkait dengan keselamatan lebih sering disebabkan oleh kesalahan dalam penerapan — penerapan — persyaratan.)

Singkatnya, kesulitan dengan persyaratan adalah akar penyebab utama dari kesalahan perangkat lunak terkait keselamatan yang telah bertahan sampai integrasi dan pengujian sistem- ing.

### Hubungan Antara Akar Penyebab dan Cacat Proses

Dalam menelusuri mundur dari kesalahan program ke sumbernya, fitur proses pengembangan sistem dapat diidentifikasi yang memfasilitasi atau memungkinkan terjadinya kesalahan. Perbedaan antara kesulitan masalah dan cara yang digunakan untuk menyelesaikannya dapat menyebabkan terjadinya kesalahan perangkat lunak yang berbahaya [4].

Langkah ketiga dari analisis kesalahan karena itu mengasosiasikan sepasang cacat proses dengan setiap kesalahan program

[16]. Elemen pertama dalam pasangan mengidentifikasi cacat proses atau ketidakcukupan dalam pengendalian kompleksitas sistem (misalnya, persyaratan yang tidak ditemukan sampai pengujian sistem). Elemen kedua dari pasangan mengidentifikasi proses yang berhubungan dengan cacat komunikasi atau de »e lo p me nl melh ods yang digunakan (misalnya, metode spesifikasi yang tepat atau tidak sistematis). Kedua elemen dari pasangan cacat proses sangat erat kaitannya. Seringkali, seperti yang akan dilihat di Sekte. V, solusi untuk satu cacat akan memberikan solusi untuk cacat terkait.

Untuk gangguan permukaan yang terkait dengan keselamatan, cacat kontrol kompleksitas yang paling umum adalah antarmuka yang tidak cukup diidentifikasi atau dipahami (56% di Voyager; 87% di Galileo). Cacat terkait keselamatan yang paling umum dalam metode komunikasi atau pengembangan yang digunakan di Voyager adalah pembelian perangkat keras no I docmm en I ed (44%). Di Galileo, kelemahan terkait keselamatan yang paling umum adalah kurangnya kamera komunik antara P / W dan S / WI (35%) dan spesifikasi antarmuka yang dipimpin atau dokumen komunik ( 35%).

*Sebuah perangkat keras yang luar biasa beha vio r* adalah faktor yang lebih signifikan dalam hal keselamatan yang terkait daripada gangguan permukaan yang tidak terkait dengan keselamatan. Ini sering dikaitkan dengan desain antarmuka selama pengujian sistem, indikasi lain dari produk perangkat lunak yang tidak stabil.

Ada variasi signifikan dalam cacat proses yang menyebabkan kesalahan antara dua pesawat ruang angkasa. Desain antarmuka selama pengujian terlibat dalam hampir seperlima dari kesalahan antarmuka keamanan kritis pada Voyager, tetapi tidak satupun dari kesalahan tersebut di Galileo. Ini karena di Voyager serangkaian masalah perangkat keras terkait menghasilkan hampir setengah dari kesalahan antarmuka terkait keselamatan. Di sisi lain, masalah spesifikasi antarmuka yang diketahui tetapi tidak didokumentasikan lebih sering terjadi di Galileo. Ini mungkin karena kompleksitas yang meningkat dari antarmuka Galileo.

Untuk kesalahan fungsional, persyaratan tidak diidentifikasi dan persyaratan tidak dipahami adalah kelemahan kontrol kompleksitas yang paling umum. Kesalahan fungsional yang berhubungan dengan keselamatan lebih mungkin terjadi daripada kesalahan fungsional yang tidak berhubungan dengan keselamatan yang disebabkan oleh re2uirerrietits trhic / t home tidak saya telah teridentifikasi.

Berkenaan dengan kekurangan dalam komunikasi atau metode pengembangan, persyaratan rnissin g dilibatkan di hampir setengah dari kesalahan terkait keselamatan yang melibatkan pengakuan persyaratan. Inodequ ate desiyn adalah cacat paling umum yang menyebabkan kesalahan dalam menerapkan persyaratan di Voyager. Di Galileo, persyaratan dokumen yang tidak lengkap sama pentingnya dengan faktor kesalahan terkait keselamatan, tetapi tidak untuk kesalahan yang tidak terkait dengan keselamatan.

***Tidak tepat atau tidak sehat*** spesifikasi lebih dari dua kali lebih mungkin untuk dikaitkan dengan kesalahan fungsional yang berhubungan dengan keselamatan dibandingkan dengan kesalahan fungsional yang tidak berhubungan dengan keselamatan. Demikian pula, persyaratan yang tidak diketahui, tidak terdokumentasi, atau buruk adalah penyebab yang lebih besar dari kesalahan terkait keselamatan daripada kesalahan yang tidak terkait dengan keselamatan.

Hasil ini menunjukkan bahwa sumber kesalahan perangkat lunak terkait keselamatan terletak jauh di belakang dalam proses pengembangan perangkat lunak — dalam persyaratan yang tidak memadai — sedangkan sumber kesalahan yang tidak terkait keamanan lebih sering melibatkan ketidakcukupan dalam fase desain.

# Perbandingan Hasil dengan Pekerjaan Sebelumnya

Meskipun kesalahan perangkat lunak dan penyebabnya telah dipelajari secara ekstensif, pekerjaan saat ini berbeda dari sebagian besar penyelidikan sebelumnya dalam empat cara berikut:

1. Perangkat lunak yang dipilih untuk dianalisis di sebagian besar studi tidak tertanam dalam sistem yang kompleks seperti yang ada di sini. Konsekuensinya adalah peran spesifikasi antarmuka dalam mengendalikan bahaya perangkat lunak telah diremehkan.
2. Tidak seperti makalah saat ini, sebagian besar studi telah menganalisis sistem yang cukup sederhana dalam domain aplikasi yang sudah dikenal dan dipahami dengan baik. Akibatnya, beberapa kesalahan perangkat lunak telah terjadi selama pengujian sistem di sebagian besar studi, yang menyebabkan kesenjangan dalam pengetahuan mengenai sumber kesalahan yang lebih persisten dan seringkali lebih berbahaya ini.
3. Kebanyakan studi berasumsi bahwa spesifikasi persyaratan sudah benar. Pada pesawat ruang angkasa, seperti dalam banyak sistem yang besar dan kompleks, persyaratan berkembang seiring dengan pengetahuan perilaku sistem dan masalah utama berkembang. Demikian pula, sebagian besar studi berasumsi bahwa persyaratan ditetapkan pada saat pengujian sistem dimulai. Ini mengarah pada meremehkan dampaknya

persyaratan yang tidak diketahui tentang ruang lingkup dan jadwal tahap selanjutnya dari proses pengembangan perangkat lunak.

1. Perbedaan antara penyebab kesalahan perangkat lunak yang kritis terhadap keselamatan dan yang tidak kritis terhadap keamanan belum diselidiki secara memadai. Upaya untuk meningkatkan keamanan sistem dengan secara khusus menargetkan penyebab kesalahan terkait keselamatan, dan dibedakan dari penyebab semua kesalahan, dapat memanfaatkan mekanisme kesalahan yang berbeda, seperti yang dijelaskan dalam Bagian. 5.

Penjelasan singkat tentang ruang lingkup dan hasil dari beberapa pekerjaan terkait diberikan di bawah ini dan dibandingkan dengan hasil yang disajikan dalam makalah ini untuk sistem komputer tertanam yang kritis terhadap keselamatan.

Nakajo dan Kume mengkategorikan 670 kesalahan yang ditemukan selama pengembangan perangkat lunak dari dua produk firmware untuk mengontrol alat ukur dan dua produk perangkat lunak untuk program pengukuran alat.

{16]. Lebih dari 90% kesalahan adalah kesalahan antarmuka atau fungsional, mirip dengan hasil yang dilaporkan di sini. Sementara kesalahan manusia utama pada pesawat ruang angkasa melibatkan komunikasi antar tim, kesalahan manusia utama dalam studi mereka melibatkan komunikasi dalam tim pengembangan. Akhirnya, cacat proses utama yang mereka identifikasi adalah kurangnya metode untuk merekam antarmuka yang diketahui dan menjelaskan fungsi yang diketahui. Dalam perangkat lunak yang disematkan pada keamanan kritis, pada pesawat ruang angkasa, cacat lebih sering adalah kegagalan untuk mengidentifikasi atau memahami persyaratan.

Ostrand dan Wey uker mengkategorikan 173 kesalahan yang ditemukan selama pengembangan dan pengujian sistem editor [19]. Hanya 2% kesalahan yang ditemukan selama pengujian sistem, yang mencerminkan kesederhanaan dan stabilitas antarmuka dan persyaratan. Sebagian besar kesalahan (61%) ditemukan selama pengujian fungsi. Lebih dari setengah dari kesalahan ini disebabkan oleh kelalaian, mengkonfirmasikan temuan dari penelitian ini bahwa kelalaian adalah penyebab utama kesalahan perangkat lunak.

Schneidewind dan Hoffmann [21] mengkategorikan 173 kesalahan yang ditemukan selama pengembangan empat program kecil oleh seorang programmer tunggal. Sekali lagi, tidak ada antarmuka yang signifikan dengan perangkat keras dan sedikit pengujian sistem. Kelas kesalahan yang paling sering terjadi, selain pengkodean dan klerikal, adalah kesalahan desain. Ketiga kesalahan desain yang paling umum — kondisi ekstrem terabaikan, kasus atau langkah yang terlupakan, dan kesalahan kontrol loop — juga merupakan kesalahan fungsional umum pada pesawat luar angkasa.

Kedua temuan yang disajikan dalam [19, 21] dan dalam makalah ini mengkonfirmasi pengalaman umum bahwa intervensi awal dan penemuan yang terlambat dari kesalahan soft wale memaksimalkan waktu dan upaya yang diperlukan untuk koreksi. Kesalahan dimasukkan dalam persyaratan dan fase desain

lebih lama untuk ditemukan dan dikoreksi daripada yang dimasukkan dalam fase selanjutnya (karena mereka cenderung melibatkan struktur perangkat lunak yang kompleks). Kesalahan yang ditemukan dalam tahap pengujian membutuhkan waktu lebih lama untuk diperbaiki (karena cenderung lebih rumit dan sulit diisolasi). Ini konsisten dengan hasil dalam [18] yang menunjukkan bahwa kesalahan yang lebih parah membutuhkan waktu lebih lama untuk ditemukan daripada kesalahan yang kurang parah selama pengujian tingkat sistem. Selain itu, efek ini ditemukan lebih jelas dalam perangkat lunak yang lebih kompleks (diukur dengan baris kode).

Pekerjaan yang dilakukan oleh Endres adalah pendahulu langsung dari Nakajo dan Kume di mana Endres mundur dari jenis kesalahan ke penyebab teknis dan organisasi yang menyebabkan setiap jenis kesalahan [4]. Selain itu, karena dia mempelajari pengujian sistem dari suatu sistem operasi, interaksi perangkat lunak dengan perangkat keras menjadi sumber perhatian. Endres mencatat kesulitan untuk secara tepat menentukan tuntutan fungsional pada sistem sebelum programmer melihat efeknya pada perilaku dinamis sistem. Kesimpulannya bahwa alat yang lebih baik diperlukan untuk mengatasi masalah ini masih berlaku delapan belas tahun setelah dia menerbitkan studinya.

Dari 432 kesalahan yang dianalisis Endres, 46% adalah kesalahan dalam memahami atau mengkomunikasikan masalah, atau dalam pemilihan solusi, 38% adalah kesalahan dalam menerapkan solusi, dan 16% sisanya adalah kesalahan cc. Hasil ini konsisten dengan temuan di sini bahwa perangkat lunak dengan banyak antarmuka sistem menampilkan persentase kesalahan perangkat lunak yang lebih tinggi dalam memahami persyaratan pemahaman atau implikasi sistem dari solusi alternatif.

Eckhardt et al., Dalam studi redundansi perangkat lunak, menganalisis kesalahan dalam dua puluh versi independen dari komponen perangkat lunak sistem navigasi inersia [3]. Dia menemukan bahwa pemahaman yang tidak memadai tentang spesifikasi atau sistem koordinat yang mendasari adalah kontributor utama kesalahan program yang menyebabkan kegagalan yang tidak disengaja.

Addy, melihat jenis kesalahan yang menyebabkan masalah keselamatan dalam sistem kontrol real-time yang besar, menyimpulkan bahwa kompleksitas desain yang melekat pada • sistem tersebut memerlukan antarmuka tersembunyi yang memungkinkan kesalahan dalam perangkat lunak non-kritis mempengaruhi keselamatan-kritis lunak adalah (l]. Hal ini konsisten dengan hasil Selby dan Basili ketika mereka menganalisis 770 kesalahan perangkat lunak selama pembaruan alat perpustakaan [22]. Dari 46 kesalahan yang disebutkan dalam laporan masalah, 70% dikategorikan sebagai "salah "Dan 28% sebagai" hilang. "Mereka menemukan bahwa subsistem yang sangat interaktif dengan subsistem lain secara proporsional memiliki lebih banyak kesalahan daripada subsistem yang kurang interaktif.

Leveson membuat daftar asumsi umum thatMenentukan perilaku eksternal perangkat lunak sering salah untuk sistem kontrol, mengakibatkan soft- (transformasi input perangkat lunak menjadi kesalahan perangkat lunak [11]. Di antara asumsi ini adalah bahwa output) hanya masuk akal jika antarmuka antara spesifikasi perangkat lunak benar, bahwa itu adalah mungkin input sistem (misalnya, kondisi lingkungan, untuk memprediksi secara realistis transien lingkungan eksekusi perangkat lunak) dan input perangkat lunak (misalnya, mon- mentasi (misalnya, keberadaan transien), dan data itor) juga ditentukan. Demikian pula, menentukan dimungkinkan untuk mengantisipasi dan menentukan benar antarmuka — terutama pengaturan waktu dan perilaku perangkat lunak ketergantungan dalam semua keadaan yang memungkinkan. hubungan — antara keluaran perangkat lunak (misalnya bintang. Asumsi ini cenderung benar untuk identifikasi sistem sederhana) dan keluaran sistem (misalnya, menutup masalah di mana kesalahan perangkat lunak telah dianalisis untuk menutup pemindai bintang) diperlukan. [5, 10]

tanggal dan palsu untuk pesawat ruang angkasa dan sistem tertanam yang besar dan penting bagi keselamatan. Jadi, sementara studi tentang kesalahan perangkat lunak dalam sistem sederhana dapat membantu dalam memahami kesalahan internal atau beberapa kesalahan fungsional, mereka kurang membantu dalam memahami penyebab kesalahan perangkat lunak terkait keselamatan, yang cenderung melibatkan

Masalah pengembangan sistem seperti waktu (aktivitas waktu nyata, penanganan interupsi, frekuensi data sensor), kemampuan dan batasan perangkat keras (kapasitas penyimpanan, transien daya, karakteristik kebisingan), tautan komunikasi (buffer dan format antarmuka), dan lingkungan operasi yang diharapkan (suhu-

antarmuka atau pengenalan kompleks Persyaratan. ture, tekanan, radiasi) perlu direfleksikan dalam

Demikian pula, ukuran standar dari spesifikasi kebutuhan perangkat lunak internal karena mereka kerumitan modul memiliki kegunaan yang terbatas di antisering sumber kesalahan perangkat lunak antarmuka perangkat lunak kritis keselamatan selama pengujian sistem. Ini bukan kesalahan.

kompleksitas internal modul tetapi kompleks-

ity dari koneksi modul ke lingkungannya yang menghasilkan kesalahan terus-menerus terkait keselamatan yang terlihat pada sistem tertanam di sini (8].

# Kesimpulan

### Rekomendasi

##### Hasil di Sect. III menunjukkan bahwa terkait dengan keselamatan

kesalahan perangkat lunak cenderung dihasilkan oleh er-

Pengaturan waktu merupakan sumber yang sulit dari kesalahan antarmuka perangkat lunak terkait keselamatan karena masalah pengaturan waktu sering kali menjadi bagian integral dari kebenaran fungsional sistem tertanam yang kritis terhadap keselamatan. Ketergantungan waktu (misalnya, berapa lama data input valid untuk membuat keputusan kontrol) harus disertakan dalam spesifikasi antarmuka perangkat lunak. Model atau simulasi analitik untuk memahami antarmuka sistem sangat berguna untuk sistem tertanam yang kompleks.

*2. Tentukan hozords yang aman dan kritis sejak awal jika dibutuhkan-*

f73 Cft US d ft 0 / §I'St9.

mekanisme yang lebih baik daripada perangkat lunak yang tidak terkait dengan keselamatan er-Bahaya ini adalah kendala di kemungkinan de- rors. Artinya, keamanan sistem bisasecara langsung tanda dan faktor dalam pikiran apapun pengorbanan ditingkatkan dengan menargetkan penyebab terkait keselamatan er-tween safety (yang cenderung mendorong simulator perangkat lunak. Secara khusus, enam berikut inirekomendasi aplikasi) dan peningkatan fungsionalitas (yang cenderung muncul dari analisis kami tentang kesalahan terkait keselamatan tidak mendorong kompleksitas perangkat lunak) [10, 22]. Banyak kompleks, tertanam kesalahan perangkat lunak terkait keamanan sistem. dilaporkan di Sect. saya II melibatkan objek data atau proses yang akan ditargetkan

*1. Fokus pada antarmuka antara soflvare dan* Perhatian khusus untuk menggunakan sistem teknik deteksi bahaya dalam menganalisis masalah tidak ada, karena SuCh ini seperti yang dijelaskan dalam [7, 11]. Antarmuka deteksi dini adalah sumber utama relokasi-keselamatan ›fl zz zz dari objek-objek keselamatan-kritis ini dan peningkatan perhatian t pp.untuk operasi perangkat lunak yang melibatkan mereka mungkin mencegah

Tujuan tradisional dari kesalahan perangkat lunak terkait keselamatan analisis persyaratan yang melibatkan mereka. fase adalah spesifikasi eksternal perangkat lunak

antarmuka ke pengguna. Definisi ini adalahtidak memadai *Gunakan spesifikasi bentuk al* fecâaiqti es sebagai tambahan

saat perangkat lunak tertanam dalam sys- ° yang lebih besar *^ \*! H T'OI- Persyaratan sosial yang terbatas*speci / ce- terns seperti pesawat ruang angkasa, pesawat canggih, trafik udara *tions.*

unit kontrol, atau kontrol proses manufaktur facil-Kurangnya presisi dan persyaratan yang tidak lengkap menyebabkannya. Dalam sistem seperti itu, perangkat lunak sering kali mengalami banyak kesalahan perangkat lunak terkait keselamatan yang terlihat sini. didistribusikan secara logis dan logis di antara berbagai perangkat keras Detail yang cukup diperlukan untuk mencakup semua keadaan komponen itu system.can dibayangkan (kegagalan komponen, waktu menipu-

pelanggaran straint, data kadaluarsa) serta validasi dokumen dan metode pengujian seperti yang dijelaskan semua asumsi lingkungan (misalnya, seberapa dekat untuk thein [9, 11] menawarkan lebih banyak cakupan.

sun instrumen akan menunjukkan) dan asumsi tentang

bagian lain dari sistem (transfer maksimum rate, fi. *Sertakan persyaratan untuk 'de termenung design "* [ITU]. konsekuensi dari kondisi ras ataucycle slippage). Banyak kesalahan perangkat lunak terkait keselamatan melibatkan Kemampuan untuk menggambarkan peristiwa dinamis, itu tanggapan perangkat lunak yang tidak memadai terhadap kondisi ekstrim dari interaksi proses yang berbeda komputer, desen - ‹› r nilai ekstrim. Perilaku perangkat keras yang tidak wajar, fungsi pengawasan yang terstruktur, dll., Seharusnya menjadi keadaan yang tidak diantisipasi, peristiwa rusak, dan usang dalam memilih metode formal [2, 4, 5, 15, 20, 23]. ‹Lata adalah penyebab kesalahan perangkat lunak yang berhubungan dengan keselamatan di

t lxe pesawat ruang angkasa.

*/. Promosi dalam komunikasi formal*mereka berakhir Ienrris. Pemeriksaan keamanan waktu jalan pada validitas memasukkan

Banyak kesalahan perangkat lunak yang terkait dengan keselamatan diakibatkan oleh data, pengatur waktu pengawas, pengatur waktu penundaan, file perangkat lunak- kesalahpahaman individu atau tim a memerlukan- hirs, yang dipaksakan perangkat lunak kondisi inisialisasi, menambahkan atau tidak mengetahui fakta tentang sistem yang menangani pengecualian sementara, dan memeriksa pernyataan yang diketahui oleh anggota tim pengembangan lain. Yang dapat digunakan untuk memerangi banyak soft-goal yang kritis terhadap keselamatan adalah untuk dapat memodularisasi tanggung jawab secara de- kesalahan ware yang melibatkan proyek velopment kesalahan bersyarat dan kelalaian tanpa modularisasi komunikasi- [11]. Persyaratan untuk penanganan kesalahan, proyeksi overflow tentang sistem yang sedang dikembangkan. Identifikasi, batas saturasi sinyal, detak jantung dan fungsi denyut nadi dan pelacakan bahaya keselamatan dalam sistem, untuk frekuensi, durasi acara maksimum, dan sistem misalnya, jelas paling baik dilakukan melintasi batas tim. havior dalam kondisi tak terduga dapat ditambahkan dan

ditelusuri ke desain. Banyak fungsional yang berhubungan dengan keselamatan

*5. Sebagai* Persyaratan euolu e, comrnunicefe t / te perubahan f: <u1ts melibatkan kesalahan-pemulihan rutinitas yang dipanggil ke dalam t / te derefoprtietif dan I est featns.a pp dengan tepat karena nilai batas yang salah atau buruk

Ini lebih penting (karena ada lebih banyak persyaratan yang berubah selama desain dan pengujian) dan lebih sulit (karena jumlah dan ukuran tim dan lamanya proses pengembangan) dalam sistem tertanam yang besar daripada dalam sistem yang lebih sederhana. Dalam menganalisis kesalahan perangkat lunak yang terkait dengan keselamatan, terlihat bahwa penentuan siapa yang perlu mengetahui tentang perubahan sering kali dibuat secara tidak tepat. Seringkali, perubahan yang tampaknya hanya melibatkan satu tim atau komponen sistem akhirnya mempengaruhi tim atau komponen lain di kemudian hari (terkadang sebagai akibat dari perubahan yang tidak kompatibel di unit yang berbeda).

Ada juga kebutuhan untuk distribusi yang lebih cepat dari perubahan yang telah dibuat, dengan pembaruan disimpan agar dapat diakses dengan ujung jari. Alat CASE menawarkan solusi yang mungkin untuk kesulitan menyebarkan perubahan tanpa menambah dokumen.

d ata.

Analisis mundur dari kegagalan kritis hingga kemungkinan penyebab menawarkan satu pemeriksaan tentang seberapa defensif persyaratan dan desain yang 12]. Spesifikasi persyaratan yang memperhitungkan skenario kasus terburuk, model yang dapat memprediksi berbagai kemungkinan (daripada yang diizinkan) kejahatan, dan simulasi yang dapat menemukan gangguan tak terduga sebelum pengujian sistem berkontribusi pada pertahanan sistem terhadap bahaya.

1. **Ringkasan dan Pekerjaan Masa Depan**

Dalam skala besar, sistem tertanam seperti dua pesawat luar angkasa dalam studi ini, kebutuhan perangkat lunak berubah selama proses pengembangan perangkat lunak, bahkan selama pengujian sistem. Hal ini sebagian besar disebabkan oleh perilaku yang tidak terduga, perubahan dinamis dalam lingkungan operasi, dan perangkat lunak / perangkat keras yang kompleks dan perangkat lunak

Prevalensi kesalahan perangkat lunak terkait keselamatan di- z adalah / interaksi perangkat lunak dalam sistem yang sedang dikembangkan-

volving salah paham atau poin persyaratan hilang

up ketidakcukupan pemeriksaan konsistensi persyaratan dan kode sebagai cara untuk menunjukkan kebenaran sistem (10]. Kode yang mengimplementasikan ulang yang salah

ot ›ed. Mengontrol perubahan persyaratan (dan, karenanya, ruang lingkup dan biaya pengembangan) sulit karena perubahan tersebut sering kali didorong oleh pemahaman yang lebih baik dari antarmuka yang diperlukan perangkat lunak dengan

persyaratan tidak benar jika gagal menyediakan dibutuhkan komponen fisik pesawat ruang angkasa yang di dalamnya

sistem tingkah laku. tertanam. Masalah waktu yang kompleks dan id perangkat keras

Demikian pula, menghasilkan kasus uji dari Nkrasi yang salah paham sering kali meminta perubahan pada persyaratan atau atau persyaratan yang hilang tidak akan menguji sistem yang benar untuk desain solusi.

ness. Ketertelusuran persyaratan dan otomatistes Analisis yang disajikan di sini tentang sebab / akibat generasi dari spesifikasi hanya menawarkan penilaian parsial dari kesalahan perangkat lunak yang terkait dengan keselamatan dengan tepat menunjukkan idation dari sistem yang kompleks dan tertanam. Aspek alternatif dari kompleksitas sistem yang bermanfaat tambahan

perhatian. Secara khusus, hasilnya telah ditunjukkanitu [9] JC Knight, "Pengujian, 'di Aerospace SoJtwore ñn-

kesalahan bersyarat (misalnya, kondisi atau nilai batas) sangat berkorelasi dengan kesalahan perangkat lunak terkait keselamatan. Kesalahan pengoperasian (terutama kelalaian waktu proses

qineerinp, Kumpulan Konsep. Ed. C. Anderson dan M. Dorfman. Washington: AIAA, 1991, hlm. 135—159.

pemeriksaan kewajaran pada data) juga sangat tinggi sesuai- [10] NG Leveson, "Keselamatan," di Aerospace SoJtwa re En-

terkait dengan kesalahan perangkat lunak terkait keselamatan. Persyaratan yang tidak diketahui, tidak berdokumen, atau salah sering dikaitkan dengan kesalahan perangkat lunak terkait keselamatan seperti

baik. Antarmuka perangkat keras / perangkat lunak telahditampilkan [11] menjadi tempat yang sering bermasalah karena kurangnya komunikasi antar tim.

***gineering,*** *Bagian Oollec dari C! O ncepts.* Ed. C. Anderson dan M. Dortman. Washington: AIAA, 1991, hlm.319—336.

NG Leveson, “Keamanan Perangkat Lunak dalam Sistem Komputer Tertanam,” C'omm un A OM, Vol. 34, No. 2, Feb 1991, hlm. 35—46.

Hasil yang disajikan dalam makalah ini menunjukkan a membutuhkan [i2] NG Leveson dan PR Harvey, 'Menganalisis Soft-

untuk metode yang lebih baik dalam menghadapi masalah dunia nyata dalam mengembangkan perangkat lunak yang sangat penting untuk keselamatan, di a

kompleks, sistem terdistribusi. Pekerjaan masa depan akandi- [i3)

##### tertarik pada menggabungkan pengetahuan dari er- yang berbeda

mekanisme ror yang menghasilkan perangkat lunak terkait keselamatan

ware Safety, "IEEE T'ronsoctfOfls On Softina re Engi-

neering, SE-9, 5, Sept 1983, hlm.569-579.

Karan L'Heureux, "Program Keamanan Sistem Perangkat Lunak RTO P, Laporan Fase A," Dokumen Internal, Laboratorium Propulsi Jet, 19 April 1991.

kesalahan ke dalam analisis persyaratan dan proses validasi. Pekerjaan diperlukan untuk menentukan bagaimana hasil ini dapat digunakan untuk memprediksi lebih tepat fitur atau kombinasi faktor apa dalam sistem tertanam yang kritis terhadap keselamatan yang mungkin menyebabkan kesalahan perangkat lunak yang memakan waktu dan berbahaya.

### Referensi

[14) R. Lutz, “Menganalisis Kesalahan Kebutuhan Perangkat Lunak dalam Keselamatan-Kritis, Sistem Tertanam,” TR92-2T, Dept. of Comp. Sci, Universitas Negeri Iowa.

1. R. Lutz dan JSK Wong, “Mendeteksi Jadwal Pemulihan Kesalahan yang Tidak Aman,” IEEE TPOns SOJtma re Eng, 18, 8, Agustus 1992, hlm. 749 — T60.
2. T. Nakajo dan H. K ume, "Analisis Kasus Sejarah Hubungan Sebab-Akibat Kesalahan Perangkat Lunak," IEEE Trans Softu ›a re Eng 17, 8, Agustus 1991, hlm. 830-838.
   1. EA Addy, “Studi Kasus tentang Isolasi Perangkat Lunak Kritis Keamanan,” dalam Prot 6 A nnuof 6'on / pada Komputer A ssura beras. N IST / IEEE, 1991, hlm.75—83.

(ITU)

PG Neuman n, “Risiko Terkait Komputer Tahun Ini: Tautan Lemah dan Peristiwa Terkait,” dalam Proc 6th Aurinal Oo nJ on Computer Assurance. NIST / IEEE, 1991, hlm. 5—8.

* 1. AM Davis, Persyaratan Perangkat Lunak, Anatisis *‹Ind* [i oleh

*Spesifikasi.* Englewood Cliifs, NJ: Prentice H semua,

1990.

DE Eckhardt, dkk., 'Evaluasi Eksperimental

AP Nikora, “Tingkat Penemuan Kesalahan berdasarkan Kategori Keparahan dan Waktu untuk Memperbaiki Kegagalan Perangkat Lunak untuk Tiga Proyek Penerbangan JPL,” Dokumen Internal, Laboratorium Propulsi Jet, 1991.

Redundansi Perangkat Lunak sebagai Strategi untuk Meningkatkan [} 9] TJ Ostrand dan EJ Wey u ker, 'Collecting and

Keandalan, "IEEE Trans Software Eng, 17, 7, Juli 1991, hlm. 692—702.

{4) A. Endres, “Analisis Kesalahan dan Penyebabnya

dalam Program Sistem, ”IEEE Trans 5oJtwore *Eng,* [20] SE-1, 2, Juni 1975, hlm.140—149.

 EM Gray dan RH Thayer, "Persyaratan," dalam Rekayasa Perangkat Lunak Dirgantara, AC! Ollection of

*Konsep s.* Ed. C. Anderson dan M. Dorfman.Mencuci- [21] ington: AIAA, 1991, hlm.89-121.

(6] ANSI / IEEE Standard Glossary of Software Engi- neering Terminology New York: IEEE, 1983.

Mengkategorikan Data Kesalahan Perangkat Lunak dalam Lingkungan Industri, "The Jo urnaI oJ Systems ntld Soft- more, 4, 1984, hlm. 289—300.

*Workshop Proc Berkele y tentang Spesifikasi Temporal dan Real-Time.* Eds. PB L adkin dan FH Vogt. Berkeley, CA: Institut Ilmu Komputer Internasional, 1990, TR-90-060.

Angin NF Schneide dan H.-M. Hoffmann, “Sebuah Eksperimen dalam Pengumpulan dan Analisis Data Kesalahan Perangkat Lunak,” IEEE Trans Software Eng, SE-5, 3, Mei 1979, hlm. 276-286.

[7] MS J affe et al., 'Software Requirements Analysis for Real-Time Process-Control Systems, "i £ 6ñ Trend Software Eng, IT, 3, March 1991, hlm. 241—

[22] RW Selby dan VR Basili, “Menganalisis Struktur Sistem yang Rawan Kesalahan,” IEEE Trans Software Eng 17, 2, Februari 1991, hlm. 141—152.

[8j

258.

P. Jalote, Pendekatan Terpadu untuk SoJtwore Engi- neering. New York: Springer-Verlag, 1991.

[23j

J. M. Wing, “Pengantar Penentu pada Formal

Metode, "Chomp ute r, Vol. 23, September 1990, hlm. 8—26.