Politechnika Warszawska
Wydział Elektryczny

Kwiecień 2016

Systemy czasu rzeczywistego

Zastosowania systemów czasu rzeczywistego w wojskowości

Autor: Michał Jereczek

Spis treści

1 Wprowadzenie **3**
1.1 Podstawowe pojęcia . . . 3
1.1.1 Klasyfikacja . . . 3

2 Historia **4**
2.1 Projekt Whirlwind . . . 4
2.1.1 Pamięć ferrytowa . . . 4
2.1.2 Whirlwind assembler . . . 4
2.2 Projekt SAGE . . . 5
2.2.1 Univac 1103A . . . 5

3 Zastosowania, a języki oprogramowania **6**
3.1 Fortran . . . 6
3.1.1 Wady . . . 7
3.1.2 Zalety . . . 7
3.1.3 Przykładowy program . . . 7
3.2 CMS-2 . . . 8
3.2.1 Naval Tactical Data System . . . 8
3.3 JOVIAL . . . 9
3.3.1 SACCS . . . 9
3.4 Ada . . . 10
3.4.1 Zalety . . . 10
3.4.2 Wady . . . 11
3.4.3 Historia . . . 11
3.4.4 Przykłady systemów . . . 12
3.4.5 Wnioski . . . 13

Bibliografia **14**

1 Wprowadzenie

1.1 Podstawowe pojęcia

Aby móc rozpocząć rozważania na temat zastosowania systemów czasu rzeczywistego w wojskowości niezbędnym jest wyjaśnienie podstawowych pojęć oraz koncepcji związanych z tego typu systemami.

1. System - W najogólniejszym znaczeniu tego zagadnienia, można powiedzieć, że system jest przyporządkowaniem pewnego zbioru wejść do pewnego zbioru wyjść.

2. System czasu rzeczywistego - Jest to system, dla którego krytycznym zagadnieniem jest czas. Poprawność działania takiego systemu jest ściśle zależna od poprawnego odebrania danych wejściowych w ściśle określonych ramach czasowych. System czasu rzeczywistego powinien zapewnić przewidywalne i możliwie bezpieczne zachowanie wobec niepowodzeń.

1.1.1 Klasyfikacja

Systemy czasu rzeczywistego można klasyfikować ze względu na skutki niezmieszczenia się w ramach czasowych, ogólnie przyjęto podział na trzy kategorie:

1. System o łagodnych ograniczeniach czasowych (ang. soft) - Jest to system, w którym przegapienie nawet wielu ram czasowych nie skutkuje poważnymi konsekwencjami, co najwyżej spadkiem wydajności i frustracją użytkowników. W zastosowaniach militarnych szczególnie ważne są systemy komunikacji, gdzie podstawową funkcjonalnością nie jest jej jakość, lecz niezawodność. Przykładem jest system audio-video, gdzie strata nawet ciągu danych nie sprawia, że cel (odebranie i zrozumienie komunikatu) nie zostanie osiągnięty (przekaz straci tylko na jakości).

2. System o mocnych ograniczeniach czasowych (ang. firm) - System, w którym przegapienie ciągu okien czasowych nie spowoduje awarii systemu, jednak spowodowane tym opóźnienie w reakcji systemu może doprowadzić do katastrofy. Wojskowe systemy szybkiego reagowania potrzebują predykcji zmian pogodowych, jest to niezwykle ważne, do poprawnego zaplanowania misji. Przykładowy system, który analizuje dane z stacji meteorologicznych w wyniku zbyt dużej ilości informacji do przetworzenia poprawnie przewidzi nastąpienie

gwałtownej burzy, jednak przez opóźnienia informacja ta może zostać podana zbyt późno aby była możliwa odpowiednia reakcja.

3. System o ostrych ograniczeniach czasowych (ang. hard) - Klasa systemu, gdzie nie zmieszczenie się w określonych ramach czasowych nawet wobec jednej operacji jest katastrofalna w skutkach. Tego typu systemy są szczególnie ważne dla wojska. Przykładem są wszelakiego rodzaju systemy kontrolujące pociski rakietowe, gdzie nawet minimalne opóźnienie między naciśnięciem przycisku do wystrzału, a samą operacją wystrzelenia może doprowadzić do nie trafienia w cel.

2 Historia

2.1 Projekt Whirlwind

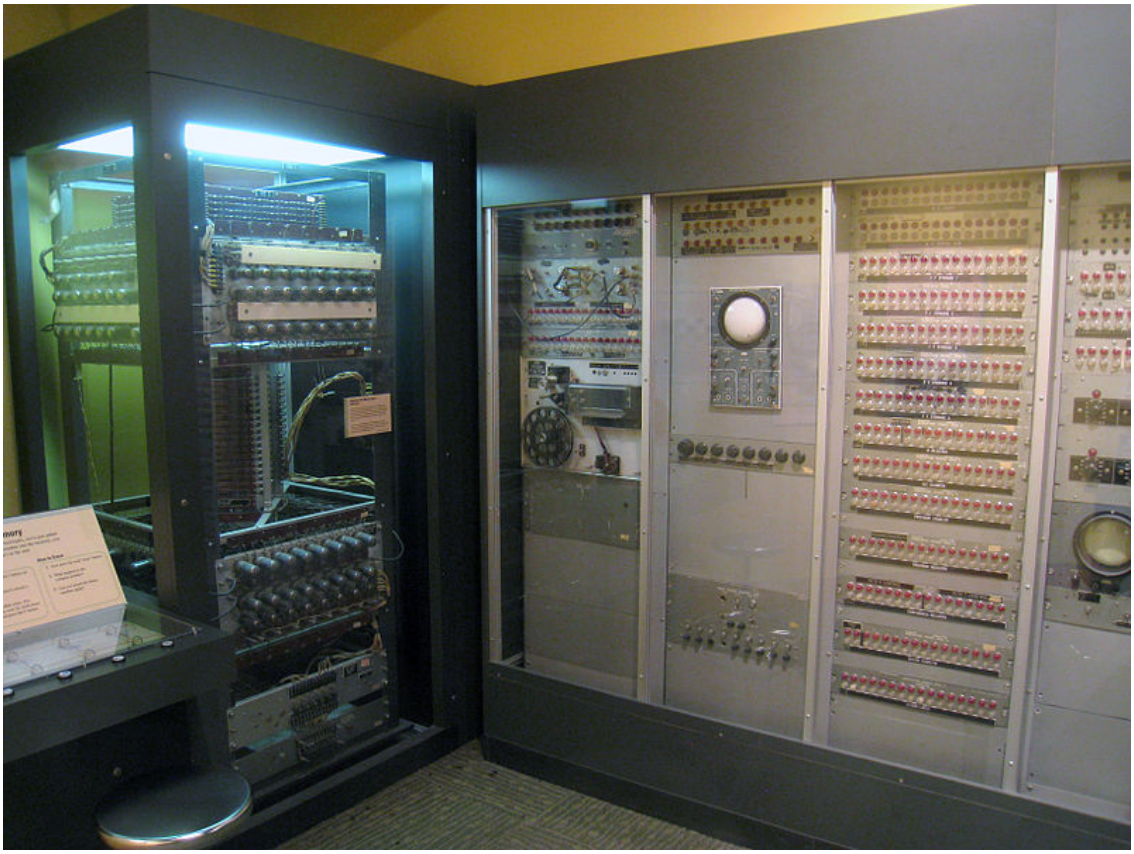
Niepewnym jest gdzie i kiedy narodził się termin *systemów czasu rzeczywistego*, jednak prawdopodobne jest, że pierwszy raz został on użyty w projekcie *Whirlwind* (pl. Trąba Powietrzna) w 1947 roku. Whirlwind to komputer wykorzystujący lampy elektronowe, który miał służyć do symulowania lotów dla załogi samolotów bombowych. Jest to jeden z pierwszych komputerów, który umożliwiał przetwarzanie równolegle w czasie rzeczywistym.

2.1.1 Pamięć ferrytowa

Whirlwind został opracowany w czasach zimnej wojny przez MIT dla marynarki wojennej Stanów Zjednoczonych. Innowacyjnym w projekcie było zastosowanie pamięci opartej o rdzeń ferrytowy, dzięki czemu dostęp do pamięci był praktycznie natychmiastowy (prekursor współczesnych pamięci RAM).

2.1.2 Whirlwind assembler

W trakcie tworzenia tego projektu utworzony został również specjalny język programowania *Whirlwind assembler*, który opisano jako *algebraiczny kompilator do uproszczenia kodowania* (ang. algebraic compiler to simplify coding), był to jeden z pierwszych języków wysokiego poziomu stworzony jeszcze na 10 lat przed Fortranem.



Rys. 1: Zdjęcie komputera Whirlwind I. Z lewej strony można zobaczyć pamięć ferrytową oraz stanowisko operatora. Źródło: `https://en.wikipedia.org/wiki/Whirlwind_I`

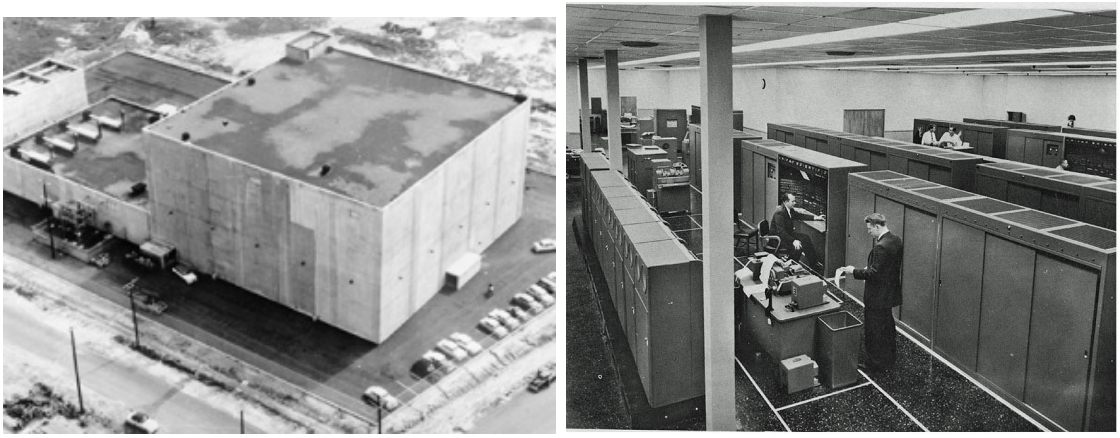
2.2 Projekt SAGE

W 1957 roku (10 lat po projekcie Whirlwind) swoje piętno w historii Sił Powietrznych Ameryki odcisnął projekt SAGE: Semi-Automatic Ground Environment. Inżynierowie IBM w współpracy z MIT stworzyli olbrzymi system komputerów, którego zadaniem było między innymi koordynowanie i przetwarzanie danych przyjmowanych z wielu systemów radarowych. System ten w rezultacie był w stanie przedstawić wizualizacje ruchu powietrznego w badanych obszarach.

2.2.1 Univac 1103A

Zadania wykonywane przez system SAGE nie były by możliwe bez użycia nowoczesnych na tamte czasy komputerów IBM'u *Univac Scientific 1103A*, które obsługiwały przerwania asynchroniczne. Komputery potrzebne do pracy systemu zajmo-

wały specjalnie postawiony budynek o powierzchni 1.4 hektara.



Rys. 2: Budynek w którym znajdował się SAGE oraz komputer Univac.
Źródła:
`https://en.wikipedia.org/wiki/Semi-Automatic_Ground_Environment`
`https://en.wikipedia.org/wiki/UNIVAC_1103`

3 Zastosowania, a języki oprogramowania

Rozwój systemów czasu rzeczywistego wymusił potrzebę stworzenia nowych, wysokopoziomowych języków oprogramowania, które ułatwiły by pracę inżynierom. U.S. Army w swoich systemach stosowało głównie język *Fortran*, U.S. Navy natomiast *CMS-2*, gdy w tym samym czasie Air Force wykorzystywało *JOVIAL*.

Tak duży podział wśród tych jednostek doprowadził do tego, że w latach siedemdziesiątych Departament Obrony Stanów zjednoczonych zlecił opracowanie języka Ada, który miał być standardowo stosowany przez nowe systemy tworzone na zlecenie wyżej wymienionych jednostek.

3.1 Fortran

Fortran(Formula Translator) jest najstarszym z języków wysokiego poziomu stosowanym współcześnie w systemach czasu rzeczywistego. Stworzony przez IBM dominował na arenie programów służących do analizowania zmian pogodowych, analiz fizycznych i chemicznych oraz innych obszarów, gdzie priorytetowa była wydajność i responsywność.

3.1.1 Wady

Fortran został stworzony około roku 1955 i jego ówczesna wersja nie wspierała tak ważnych dla systemów czasu rzeczywistego funkcji jak *przerwania* i *kolejkowanie zdarzeń*, dlatego musiał współpracować razem z dużymi porcjami kodu assemblera. Język ten nie posiadał również mechanizmów dynamicznego alokowania pamięci oraz rekurencji.

3.1.2 Zalety

Fortran pozwalał na pisanie konstrukcji typu if-then-else, co umożliwiało pisanie na znacznie wyższym poziomie abstrakcji niż dotychczas. Język ten powstawał w czasach małych i wolnych systemów wbudowanych, co przyczyniło się do zaprojektowania go w taki sposób, aby można w nim było pisać wydajne programy przy małej mocy obliczeniowej.

3.1.3 Przykładowy program

Przykładowy program, który losuje numer pomiędzy 0, a 1 i sprawdza czy mieści się w określonych przedziałach.

```
program xif
   implicit none
   real :: x
   real, parameter :: x1 = 0.3, x2 = 0.6
   call random_seed()
   call random_number(x)
   if (x < x1) then
      print*,x,"<",x1
   else if (x < x2) then
      print*,x,"<",x2
   else
      print*,x,">=",x2
   end if
end program xif
```

źródło: `https://en.wikibooks.org/wiki/Fortran/Fortran_control`

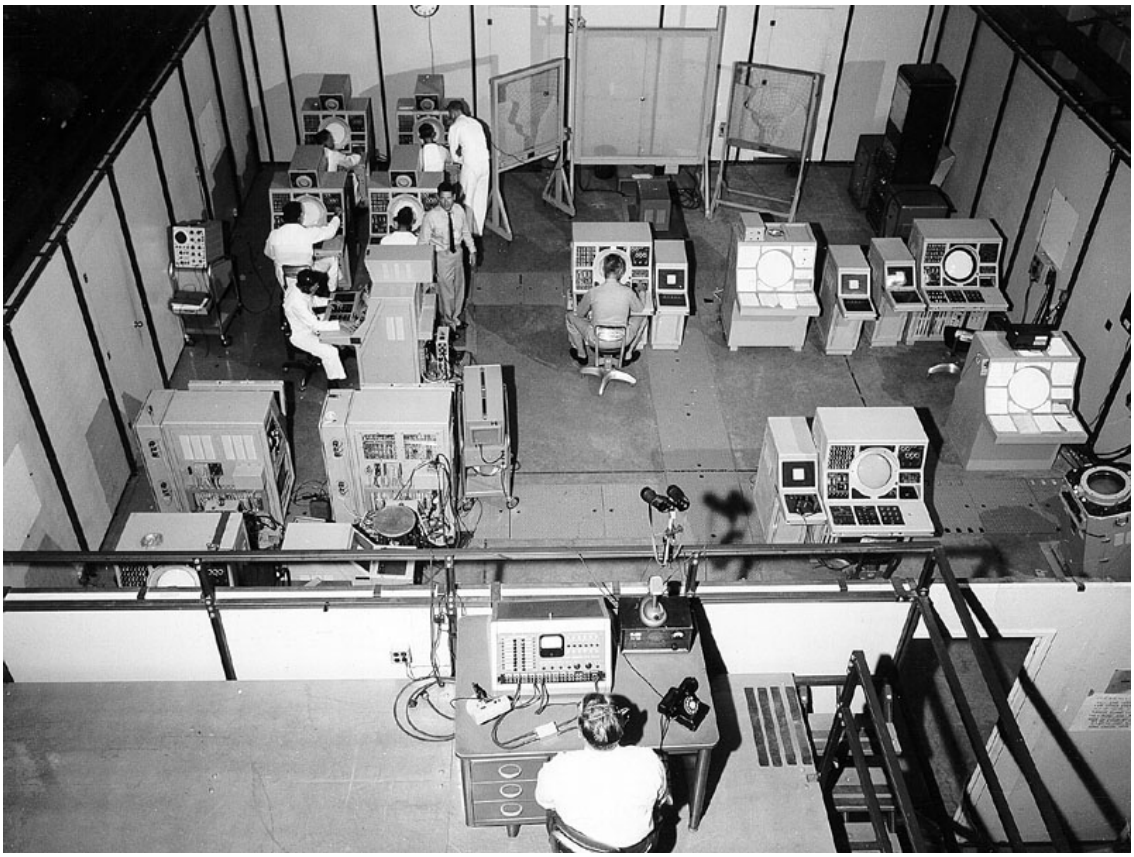
3.2 CMS-2

CMS-2 (Compiler Monitor System 2) był językiem stworzonym specjalnie na potrzeby U.S. Navy. Wydany około roku 1968 (ponad dekadę po Fortranie) stosowany jest w systemach wbudowanych.

3.2.1 Naval Tactical Data System

CMS-2 stosowany jest chociażby w systemie NTDS (Naval Tactical Data System), który wspiera misje bojowe do dnia dzisiejszego.

System ten kolekcjonuje i procesuje ważne informacje aby przedstawić aktualną sytuację, dzięki czemu dowódca sił zbrojnych może szybko podejmować strategiczne decyzje. Okręty wojenne z systemem NTDS komunikują się wzajemnie, co pozwala na zbudowanie szczegółowej mapy taktycznej.



Rys. 3: Zdjęcie przedstawiające testy systemu NTDS w zainscenizowanym pokładowym pomieszczeniu operacyjnym CIC (Combat Information Center).
Źródło: `https://en.wikipedia.org/wiki/Naval_Tactical_Data_System`

3.3 JOVIAL

JOVIAL (Jules Own Version of the International Algorithmic Language) był wysokopoziomowym językiem zaprojektowanym przez Julesa Schwartz na potrzeby militarne Stanów Zjednoczonych Ameryki.

Język powstały w 1959 roku i był odpowiedzią na brak alternatyw dla programowania systemów wbudowanych czasu rzeczywistego. US Air Force stosuje go do dnia dzisiejszego w celu programowania urządzeń z procesorami 1750A.

3.3.1 SACCS

SACCS (Strategic Automated Command Control System), potężny system do wymiany krytycznych komunikatów poprzez sieć, został w 95% napisany w języku JOVIAL.

Stworzenie systemu pochłonęło 2 lata, w tym 1400 programistycznych lat, ponad dwukrotnie mniej niż w porównywalnym w skali projekcie SAGE. Tak duże przyspieszenie zdołano uzyskać dzięki nowoczesnemu (na te czasy) językowi. SAGE był programowany w unowocześnionej odmianie Whirlwind Assemblera, która nie pozwalała na tak wysoki poziom abstrakcji jak JOVIAL.

System SACCS zapewniał komunikację między ważnymi strukturami Departamentu Obrony Narodowej takimi jak: centrale kontrolne wyrzutni rakietowych, strategiczne centra komunikacji, systemy komunikacji lotniczej, jednostki nuklearne. Innowacyjnym w systemie było zapewnienie bezpiecznej i dwukierunkowej (full-duplex) komunikacji sieciowej.

Natura SACCS jako systemu czasu rzeczywistego przejawiała się na froncie. Dowódcy sił zbrojnych za pośrednictwem systemu mogli otrzymywać i wydawać krytyczne komunikaty, system był w stanie przekazywać aktualne dane z interfejsów: DPS (Subsystemu przetwarzania danych), AFGWC (Globalnego centrum zmian pogodowych U.S. Airforce), AFSATCOM (System satelitarny U.S. Airforce) oraz 616A (System niezawodnej komunikacji niskich częstotliwości).

System gwarantował odebranie strategicznych komunikatów w ciągu 15 sekund, co w latach osiemdziesiątych było olbrzymim sukcesem, trzeba pamiętać, że komunikacja była dodatkowo szyfrowana, co znacznie spowalniało czas reakcji.

Rys. 4: Centrala kontrolna wyrzutni rakietowych. źródło: `http://fas.org/nuke/guide/usa/c3i/cvpmrjan/sld013.htm`

3.4 Ada

Język Ada został opracowany na przełomie lat osiemdziesiątych aby ostatecznie być wydanym w 1983 roku. Głównym celem, który przyświecał jego realizacji była minimalizacja kosztów związanych z utrzymaniem systemów stworzonych na potrzeby militarne Stanów Zjednoczonych oraz zapewnienie języka, który pozwoliłby pisać kod na bardzo wysokim poziomie abstrakcji, kod czytelny w odbiorze. Język miał służyć do programowania olbrzymich systemów wbudowanych, które wymagałyby mechanizmów związanych z przetwarzaniem w czasie rzeczywistym.

3.4.1 Zalety

Ada posiadał szereg cech, które ze względu na krytycznych charakter systemów, dla których był przeznaczony są bardzo istotne. Do najważniejszych z nich należą:

- Programowanie generyczne - pisanie kodu abstrahującego od typu danych.

- Mechanizm wyjątków - umożliwienie bardzo szczegółowej obsługi błędów.
- Instrukcja wyboru (ang. Switch statement) - zastąpienie wielopoziomowych instrukcji typu *jeżeli-to* czytelniejszą konstrukcją.
- Typowanie silne - Zapewnia błędy kompilacji w przypadku użycia złego typu danych dla określonych operacji, dzięki czemu można zażegnać nieprzewidywalnym zachowaniom systemu.
- Obliczenia równoległe - Ada wspiera technikę SIMD(Jedna instrukcja - Wiele danych) oraz MIMD(Wiele instrukcji - Wiele danych) z wykorzystaniem pamięci współdzielonej.

3.4.2 Wady

Ada nie wspierała paradygmatu obiektowego, ten został dodany dopiero w 1995 wraz z nową wersją nazwaną *Ada 95*, wraz z aktualizacją dodano ulepszone wsparcie dla obliczeń numerycznych oraz finansowych. *Ada 95* nie jest wstecznie kompatybilna z wcześniejszymi wersjami.

3.4.3 Historia

Od 1983 roku Ada stała się bardzo popularnym językiem wykorzystywanym nie tylko w sektorze militarnym, ale również w komercyjnych systemach bankowych lub lotniczych.

W 1987 roku Departament Obrony Stanów Zjednoczonych wydał oficjalne instrukcje:

"*Język programowania Ada powinien być jedynym, powszechnie stosowanym językiem programowania dla zasobów komputerowych Obrony(USA) wykorzystywanych w inteligentnych systemach do komunikacji i kontroli sił zbrojnych, lub jako integralna część systemów zbrojeniowych.*"

Tłumaczenie cytatu z źródła:
`http://cs.stanford.edu/people/eroberts/cs201/projects/critical-systems/military.htm`

Instrukcje te utrzymywały się w mocy przez dekadę, dopóty w roku 1997 zostały wycofane w związku z badaniami przeprowadzonymi przez Radę Badawczą Narodowej Akademii Nauk. Głównym powodem były problemy z małą popularnością języka, gdyż ten był stosowany w głównej mierze w Systemach Departamentu Obrony. Języka Ada nie uczono w szkołach, a dostęp do narzędzi i kompilatorów był ograniczony.

3.4.4 Przykłady systemów

Wiele krytycznych systemów zostało opartym na tym języku, wiele z nich funkcjonuje do dzisiaj. Poniżej prezentuję najciekawsze z wybranych:

1. **CMWS (Common Missile Warning System)**

   System służący do wykrywania zbliżających się pocisków. CMWS stosowany jest w śmigłowcach, szybowcach oraz samolotach należących do U.S. Army, charakteryzuje się ekstremalnie niską częstotliwością fałszywych alarmów oraz łatwością adaptacji.

   Zestaw składający się na system zawiera pięć sensorów o wymiarach 8x8x10cm i wadze 1kg każdy oraz jedną jednostkę sterującą o wymiarach 12x25x33cm i wadze 8 kilogramów. Jednostka sterująca na podstawie danych z sensorów w momencie wykrycia nadchodzącego pocisku komunikuje się z systemem pokładowym pilota bądź załogi. Należy zwrócić uwagę na olbrzymią niezawodność systemu, ten od 2006 roku sprawdził się w ponad milionie godzin lotów w misjach wojennych łącznie dla ponad 2000 maszyn, na które został dostarczony.

Rys. 5: Zdjęcie przedstawiające osprzęt SMWS. Źródło: `http://www.baesystems.com/en/product/anaar57-common-missile-warning-system-cmws`

2. **TSS (The Boeing Company's Training Systems & Services)**

   AdaCore (firma zajmująca się dostarczaniem komercyjnych rozwiązań dla systemów pisanych w języku Ada) uczestniczy w projektach związanych z symulacją lotów w czasie rzeczywistym. Z wykorzystaniem Ada napisane zostały symulatory dla amerykańskich maszyn takich jak: Boeing C-17, Boeing AH-64D Apache Longbow,Boeing F-15E.

3. **SSDS (Ship Self-Defense System)**

   Rozwiązania firmy *AdaCore* wykorzystane zostały również w systemie obronnym okrętów wojennych opracowanym dla U.S. Navy. Język Ada został użyty w projekcie w integracji z C i C++, zaprogramowane zostały maszyny z procesorami Intela i Systemem LynxOS x86 (LynxOS jest systemem operacyjnym specjalnie zaprojektowanym do celów przetwarzania w czasie rzeczywistym).

   SSDS integruje sygnały dostarczane przez systemy różnego rodzaju detekcji takie jak na przykład ESM (System radarowy), IFF (System identyfikujący czy wykryta jednostka jest przyjacielem lub wrogiem) i opracowuje plan działania, który (zależnie od ustawień automatyzacji) może być wysłany do operatorów aby Ci podjęli działania lub automatycznie wykona możliwe czynności (wystrzeli pocisk w stronę wroga).

3.4.5 Wnioski

Język Ada w swojej nowoczesnej odsłonie (Ada 95) stosowany jest powszechnie w systemach czasu rzeczywistego mimo, że nie jest tak popularny jak inne języki takie jak C, C++, Java, które dają podobne możliwości. Spowodowane jest to tym, że język ten od początku projektowany był z myślą o krytycznych systemach, dzięki czemu ryzyko popełnienia dotkliwych błędów jest mocno ograniczone.

O sukcesie Ada świadczy również szerokie jego wykorzystanie nie tylko w systemach armii amerykańskiej, ale również kanadyjskiej, australijskiej, szwedzkiej, brytyjskiej i innych.

Bibliografia

[1] Phillip A. Laplante, *Real-time systems design and analysis*. IEEE PRESS, Third Edition, 2004.

[2] https://pl.wikipedia.org/wiki/System_czasu_rzeczywistego

[3] https://en.wikipedia.org/wiki/Whirlwind_I

[4] http://ethw.org/Magnetic-Core_Memory

[5] https://en.wikipedia.org/wiki/Semi-Automatic_Ground_Environment

[6] https://en.wikipedia.org/wiki/UNIVAC_1103

[7] https://en.wikipedia.org/wiki/CMS-2_(programming_language)

[8] http://www.thefreedictionary.com/naval+tactical+data+system

[9] http://mil-embedded.com/articles/ntds-navy-platforms-worldwide/

[10] https://en.wikipedia.org/wiki/Naval_Tactical_Data_System

[11] http://progopedia.com/implementation/jovial/

[12] http://fas.org/nuke/guide/usa/c3i/saccs.htm

[13] http://www.ibspan.waw.pl/~paprzyck/mp/cvr/research/varia_papers/ADA_para_97.pdf

[14] http://cs.stanford.edu/people/eroberts/cs201/projects/critical-systems/military.htm

[15] http://www.seas.gwu.edu/~mfeldman/ada-project-summary.html#Military_Applications_

[16] http://www.baesystems.com/en/product/anaar57-common-missile-warning-system-cmws

[17] http://www.adacore.com/press/gnat-pro-development-environment-to-support-boeinge28099s-real-time-simulat/

[18] http://www.adacore.com/press/gnat-pro-provides-multi-language-support-aboard-raytheons-ssds