Techniki komputerowe w robotyce

WYKŁAD OSTATNI

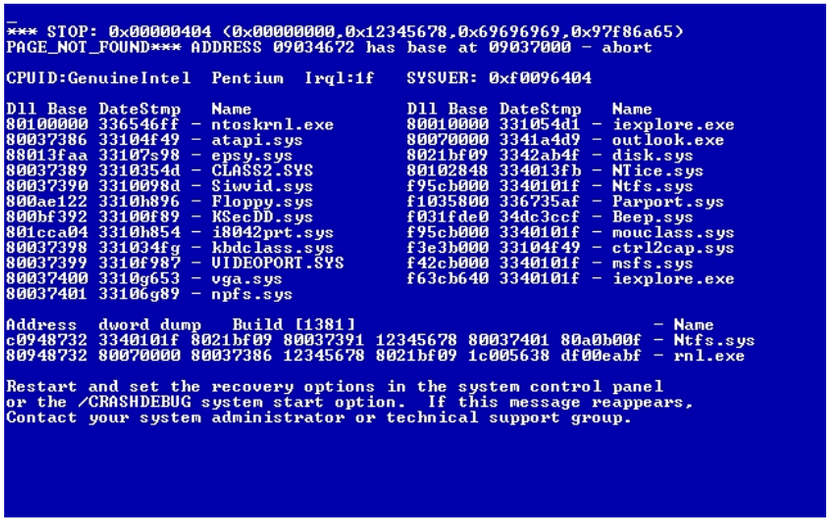
Błędy systemów komputerowych – źródła i klasyfikacja

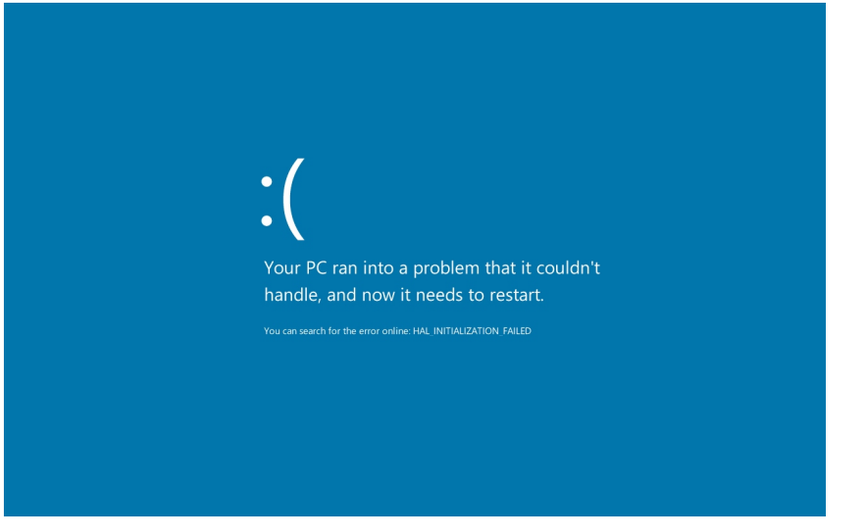
Robert Muszyński
KCiR, W4, PWr

– Skład FoilTEX –

Błędy systemów komputerowych

Błąd – właściwość programu lub sprzętu komputerowego powodująca nieprawidłowe działanie





Dążenie do opracowania systemu bezbłędnego – mrzonka

Błędy systemów komputerowych

Wymagamy by system komputerowy był:

- poprawny
- niezawodny
- bezpieczny

Poprawność – stała własność systemu, polegająca na tym, że system jest zgodny z jego specyfikacją

Niezawodność – stopień w jakim można się spodziewać, że system realizuje planowane funkcje z wymaganą precyzją

Bezpieczeństwo – zespół zasad, jakimi należy się kierować projektując i wykorzystując system komputerowy, tak aby poprawnie i w całości realizował on tylko i wyłącznie cele zgodne z intencjami właściciela

Poprawność a niezawodność

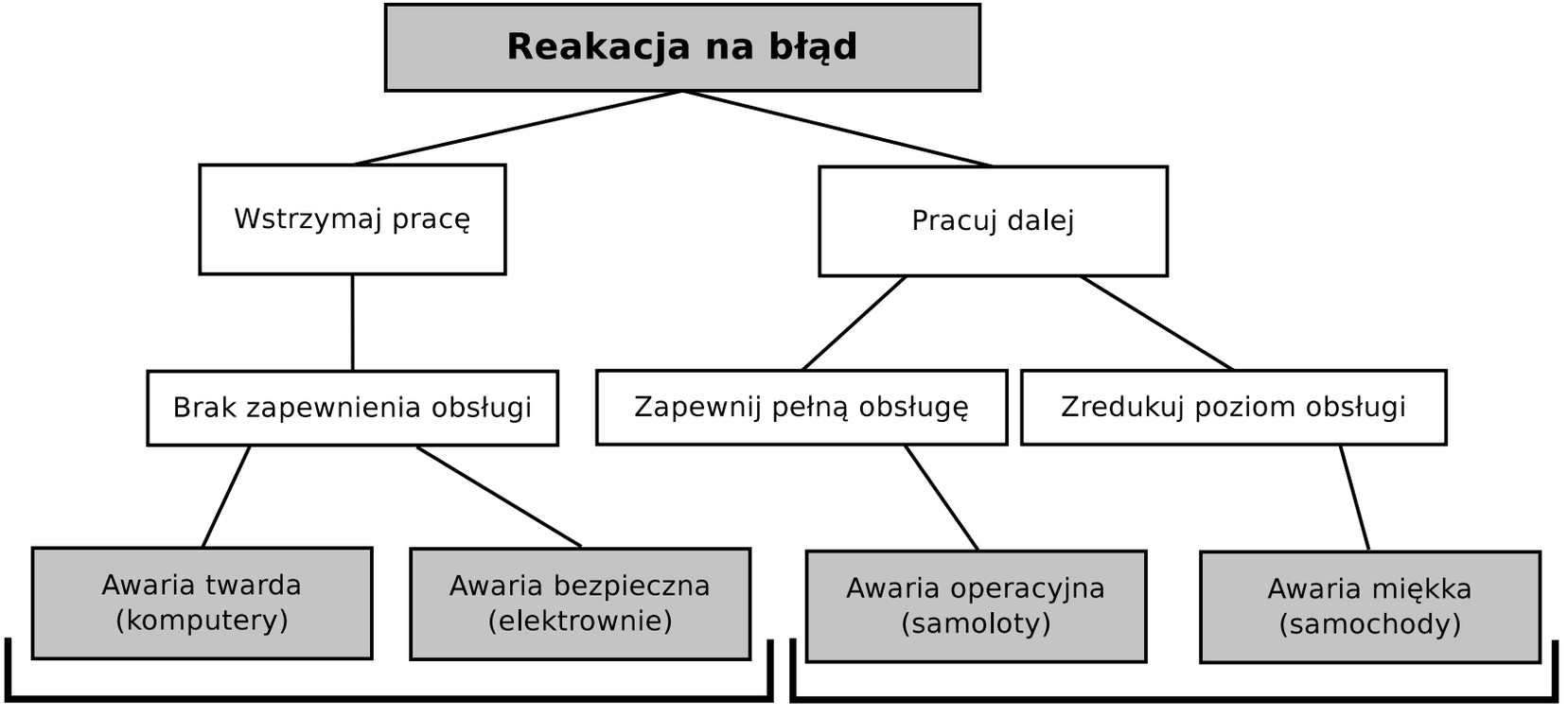
Czy system uznany za poprawny może nie być niezawodny i czy system niepoprawny może być niezawodny?

1. System sprawdzony w warunkach laboratoryjnych nie działa w miejscu docelowym
2. Źle napisany system mimo to działa niezawodnie, bo nadrzędne jego warstwy korygują działania źle napisanych warstw podrzędnych

Bezpieczeństwo a niezawodność

System w 100% bezpieczny, może być w 0% niezawodny i vice versa

Sposoby obsługi błędów



Klasyfikacja błędów

Błędy systemu:

- na etapie projektowania
- na etapie programowania
  - syntaktyczne
  - semantyczne
  - logiczne
  - algorytmiczne
- wynikające z wpływu otoczenia

Przykłady błędów

- Na etapie projektowania
  - maszty radarowe na angielskich niszczycielach
  - mechanizm zwalniania rakiet w F-18
  - HMS Sheffield – Falklandy-Malwiny
  - USS Vincennes z system AEGIS i irański Airbus A300
  - F-16 vs rakiety Patriot
- Na etapie programowania – błędy składni
  - użycie niepoprawnego symbolu (`var c:integer;`)
  - niepoprawne użycie symbolu (`x=y` zamiast `x==y`)
- Na etapie programowania – błędy znaczenia
  - niewłaściwie zrozumiana specyfikacja
  - niewłaściwie realizowana specyfikacja

Zasada

Czym mniej napiszesz, tym mniej popełnisz błędów – dobór właściwego języka programowania

Przykłady błędów

- Na etapie programowania – błędy logiczne
  - pętle z warunkami post-check zamiast pre-check
  - przypadkowe pętle nieskończone
  - niezainicjowane zmienne
  - zakleszczenia (deadlock)
  - źle sformułowane warunki logiczne
- Na etapie programowania – błędy algorytmiczne
  - przekroczenie zakresu liczb
  - przekroczenie precyzji
  - nieuwzględnienie zakresu wartości obsługiwanych przez urządzenia we/wy
- Wynikające z wpływu otoczenia

Sprzęt $\longleftrightarrow$ Oprogramowanie $\longleftrightarrow$ Człowiek

Ranking błędów (według MITRE CWE/SANS)

25. Wyścig (race condition)
24. Używanie złych algorytmów kryptograficznych
23. Otwarte przekierowania (np. phishing)
22. Nielimitowana alokacja zasobów
21. Złe uprawnienia do krytycznych zasobów
20. Pobieranie niesprawdzonego kodu
19. Brak uwierzytelnienia przy krytycznych funkcjach systemu
18. Błąd w obliczeniach długości bufora
17. Przepełnienie wartości liczby całkowitej
16. Ujawnianie informacji w komunikatach błędów
15. Zła obsługa nadzwyczajnych lub warunkowych zdarzeń/wartości
14. Zła walidacja wskaźnika
13. Brak kontroli nazwy pliku dla polecenia Include/Require w języku PHP

12. Dostęp do bufora z niewłaściwą długością
11. Osadzenie uwierzytelnienia w oprogramowaniu
10. Brak szyfrowania wrażliwych danych
9. Brak kontroli parametrów przekazywanych do systemu operacyjnego
8. Brak ograniczeń typów plików, które można umieścić na serwerze
7. Brak ograniczeń ścieżki do konkretnego katalogu (path traversal)
6. Poleganie na niezaufanych informacjach przy uwierzytelnieniu
5. Błędy w przyznawaniu uprawnień
4. Cross-site Request Forgery (CSRF)
3. Kopiowanie danych bez kontroli długości (przepełnienie bufora) – klasyk!
2. Niewłaściwa kontrola parametrów używanych do zapytania SQL
1. Brak zachowania struktury serwisu webowego (Cross-site Scripting)