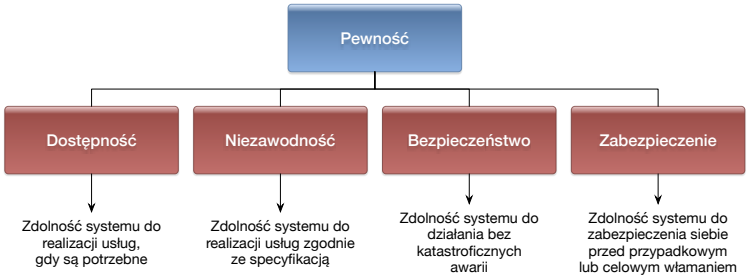
# Systemy krytyczne

- Pewność (2h)
- Specyfikowanie systemów krytycznych (1h)
- Wytwarzanie systemów krytycznych (1h)

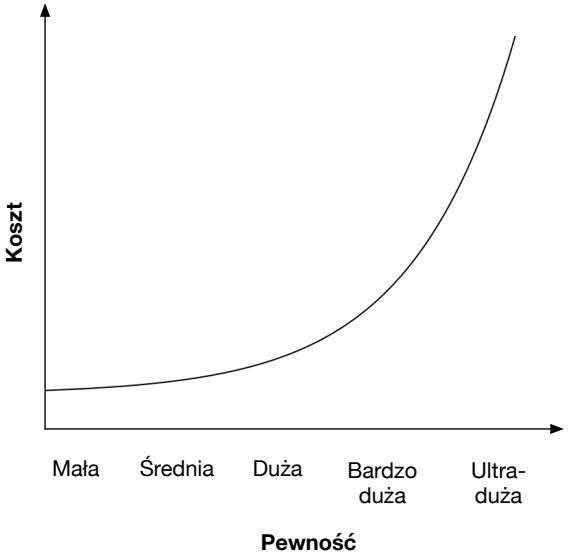



Pewność ma cztery zasadnicze wymiary:

1. **Dostępność** — prawdopodobieństwo, że w ustalonej chwili będzie on włączony, będzie działał i będzie w stanie realizować użyteczne usługi.
2. **Niezawodność** — prawdopodobieństwo, że w ustalonym okresie system będzie poprawnie realizował usługi zlecone przez użytkownika.
3. **Bezpieczeństwo** — opinia o możliwości wyrządzenia szkody ludziom i środowisku przez system.
4. **Zabezpieczenie** — opinia o odporności systemu na przypadkowe lub celowe wtargnięcie intruza.

# **Pewność**

- **Pewność** systemu komputerowego jest własnością systemu odpowiadającą zaufaniu, którym jest obdarzany.
- **Zaufanie** oznacza stopień przekonania użytkownika, że system będzie działał tak jak należy i nie będzie się „psuł” przy normalnym użytkowaniu.




Wykładniczy wzrost kosztu

# Systemy krytyczne

- **Systemy krytyczne dla bezpieczeństwa** — awaria może powodować obrażenia i utratę życia lub poważne zniszczenia środowiska.
- **Systemy krytyczne dla zadania** — awaria może powodować niepowodzenie pewnej czynności mającej jakiś cel.
- **Systemy krytyczne dla przedsiębiorstwa** — awaria może powodować kłopoty korzystających z nich przedsiębiorstw.

- Koszt awarii systemu krytycznego jest zwykle bardzo wysoki i obejmuje:
  - **koszty awarii** (np. konieczność wymiany systemu),
  - **koszty pośrednie** (koszty procesów sądowych, utracone zyski w związku z niedostępnością systemu)

- Systemy krytyczne są zwykle budowane w oparciu o starannie sprawdzone metody a nie nowinki.
- Dopiero niedawno zaczęto stosować metody obiektowe do tworzenia systemów krytycznych.
- Wiele systemów krytycznych nadal powstaje w ramach programowania funkcyjnego[1)].

[1)] Funkcyjna strategia projektowania polega na dzieleniu programu na zbiór oddziałujących funkcji lub podprogramów ze scentralizowanym stanem systemu współdzielonym przez te funkcje.

# Dostępność i niezawodność

- **Niezawodność**. Prawdopodobieństwo bezawaryjnego działania w ciągu ustalonego czasu w zadanym środowisku w określonym celu.
- **Dostępność**. Prawdopodobieństwo, że w ustalonej chwili system będzie działał i będzie zdolny do realizacji usług.
- Dostępność nie zależy od samego systemu ale od czasu niezbędnego do usunięcia usterek, które czynią go niedostępnym.

Przykład

- Jeśli system A ulega awarii raz na rok a system B raz na miesiąc, to A jest bardziej niezawodny niż B.
- Jeśli uruchomienie A wymaga jednak trzech dni, a B dziesięciu minut, to rocznie dostępność B jest istotnie większa niż A.
- Użytkownicy będą raczej woleli system B niż A.

| Pojęcie | Opis |
|---|---|
| Awaria systemu | Zdarzenie zachodzące w chwili, w której system przestaje realizować usługi oczekiwane przez użytkownika. |
| Błąd systemu | Błędne zachowane systemu niezgodne ze specyfikacją. |
| Usterka systemu | Niepoprawny stan systemu, tzn. stan, którego nie spodziewali się projektanci systemu. |
| Błąd lub pomyłka człowieka | Działanie człowieka, które powoduje usterki systemu. |

Pojęcia związane z niezawodnością

- **Unikanie usterek**. Stosuje się metody tworzenia, które zmniejszają możliwości pomyłek lub umożliwiają wychwytywanie ich, zanim doprowadzą do awarii systemu (unikanie podatnych konstrukcji języka jak wskaźniki, użycie analizy statycznej).
- **Wykrywanie i usuwanie usterek**. Stosuje się metody weryfikacji i zatwierdzania, które zwiększają szansę wykrycia i usunięcia usterek przed przystąpieniem do użytkowania systemu (systematyczne testowanie i usuwanie błędów przez śledzenie wykonywania kodu).
- **Tolerowanie usterek**. Stosuje się metody, które zapewniają, że usterki w systemie nie powodują błędów, a błędy systemu nie powodują awarii (wprowadzenie do systemu udogodnień wewnętrznej kontroli i użycie nadmiarowych modułów systemu).

# Bezpieczeństwo

- **Podstawowe oprogramowanie krytyczne dla bezpieczeństwa** jest oprogramowaniem w postaci sterownika wbudowanego w system (niewłaściwe działanie takiego oprogramowania może powodować niewłaściwe działanie sprzętu, które powoduje szkody lub zniszczenie środowiska).
- **Pomocnicze oprogramowanie krytyczne dla bezpieczeństwa** jest oprogramowaniem, które może pośrednio powodować szkody (np. niewłaściwe działanie systemu komputerowego wspomagania projektowania może doprowadzić do usterek w projektowanym obiekcie).

| Pojęcie | Definicja |
|---|---|
| Wypadek (lub nieszczęście) | Niezaplanowane zdarzenie lub ciąg zdarzeń, które powodują obrażenia lub śmierć osób, szkody w majątku lub środowisku. Maszyna sterowana komputerem raniąca swego operatora to dobry przykład wypadku. |
| Zagrożenie | Sytuacja, która może spowodować lub przyczynić się do wypadku. Awaria detektora wykrywającego przeszkody na drodze maszyny kest przykładem zagrożenia. |
| Szkoda | Miara strat powstałych w wyniku nieszczęścia. Zakres szkód może być rozmaity, od śmierci wielu osób do mniejszych obrażeń i zniszczenia majątku. |
| Waga zagrożenia | Oszacowanie największych szkód, które mogą być wynikiem konkretnego zagrożenia. Waga zagrożenia może być rozmaita, od katastroficznej (gdy ginie wielu ludzi) do niewielkiej (jedynie niewielkie szkody). |
| Prawdopodobieństwo zagrożenia | Prawdopodobieństwo wystąpienia zdarzeń powodujących zagrożenie. Prawdopodobieństwo może być dowolne, zwykle waha się od prawdopodobne (np. zagrożenie wystąpi w jednym na sto przypadków) do niewiarygodne (nie można wyobrazić sobie sytuacji, w których to zagrożenie występuje). |
| Ryzyko | Miara prawdopodobieństwa, że system spowoduje wypadek. Ryzyko szacuje się na podstawie prawdopodobieństwa zagrożenia, wago zagrożenia i prawdopodobieństwa, że zagrożenie będzie przyczyną wypadku. |

Pojęcia związane z bezpieczeństwem

- **Unikanie zagrożeń**. System jest zaprojektowany tak aby unikać zagrożeń. System tnący, którego zadziałanie wymaga od operatora naciśnięcia dwóch przycisków w tej samej chwili, wystrzega się na przykład ryzyka umieszczenia rąk operatora w zasięgu noża.
- **Wykrywanie i eliminowanie zagrożeń**. System jest zaprojektowany tak, aby wykrywać i eliminować zagrożenia, zanim doprowadzą do wypadku. System reaktora chemicznego może na przykład wykrywać nadmierne ciśnienie i otwierać zawór bezpieczeństwa w celu zmniejszenia ciśnienia, zanim dojdzie do eksplozji.
- **Ograniczanie szkód.** System może obejmować udogodnienia zabezpieczające, które ograniczają szkody będące konsekwencją wypadku. Silnik lotniczy zawiera na przykład automatyczne gaśnice. Gdy pojawi się ogień, zwykle można go ugasić, zanim stanie się zagrożeniem dla pasażerów i załogi.

# Zabezpieczenie

- **Zabezpieczenie** systemu to oszacowanie stopnia bezpieczeństwa systemu przed atakami z zewnątrz zarówno przypadkowymi jak i celowymi.
- Przykłady ataków: wirusy, nieuprawnione użycie usług systemu, nieuprawnione aktualizowanie systemu lub jego danych itd.
- Istnieją rodzaje systemów krytycznych, w których zabezpieczenie jest najważniejszym wymiarem pewności systemu:
  - systemy wojskowe
  - systemu handlu elektronicznego
  - systemy przetwarzające i wymieniające poufne dane

- Trzy typy szkód, które mogą być wynikiem ataku zewnętrznego:
  - **Zaprzestanie usług**. System może być zmuszony do przyjęcia stanu, w którym nie jest możliwe normalne realizowanie usług.
  - **Uszkodzenie programu lub danych**. Komponenty programowe mogą być podmienione w nieuprawniony sposób. Może to wpłynąć na zachowanie systemu, a zatem na niezawodność i bezpieczeństwo.
  - **Ujawnienie poufnej informacji**. Informacja przechowywana w systemie może być poufna. Atak z zewnątrz może ujawnić ją nieuprawnionym osobom.

| Pojęcie | Definicja |
|---|---|
| Odsłonięcie | Możliwość straty lub szkody w systemie komputerowym. |
| Słabość | Słaby punkt systemu komputerowego, który można wykorzystać do spowodowania strat lub szkód. |
| Atak | Wykorzystanie słabości systemu. |
| Groźba | Sytuacja, która może doprowadzić do strat lub szkód. |
| Nadzór | Miara ochrony, która zmniejsza słabość systemu. |

Pojęcia związanie z zabezpieczeniem

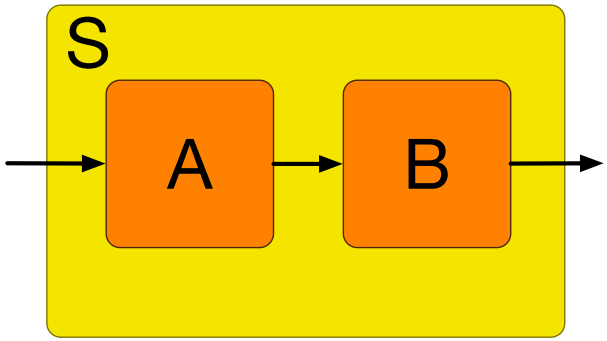
- **Unikanie słabości**. System projektuje się tak, aby nie miał słabości (np. odseparowanie systemu od sieci publicznej).
- **Wykrywanie i neutralizowanie ataków**. System projektuje się tak, by wykrywał słabości i eliminował je, zanim spowodują odsłonięcie (np. zastosowanie programów antywirusowych).
- **Ograniczanie odsłonięć**. Minimalizuje się konsekwencje udanego ataku (np. systematyczne tworzenie kopii zapasowych).

# Specyfikowanie systemów krytycznych

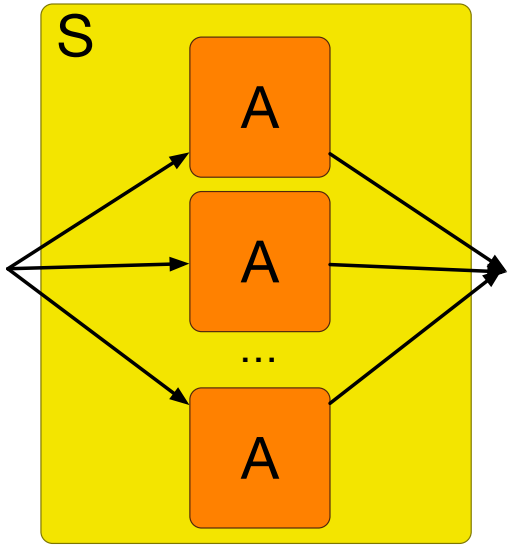
- Specyfikowanie niezawodności systemu
- Specyfikowanie bezpieczeństwa
- Specyfikowanie zabezpieczeń

## Specyfikowanie niezawodności systemu

- **Niezawodność sprzętu**. Jakie jest prawdopodobieństwo awarii komponentu sprzętowego i jak dużo czasu potrzeba na jej usunięcie?
- **Niezawodność oprogramowania**. Jakie jest prawdopodobieństwo wytworzenia niepoprawnego wyniku przez komponent programowy? (oprogramowanie się nie zużywa i może kontynuować poprawne działanie po wygenerowaniu niepoprawnych danych wyjściowych)
- **Niezawodność operatora**. Jakie jest prawdopodobieństwo popełnienia błędu przez operatora.




$P_S = P_A + P_B$




$P_S = P_A^n$

| Miara | Objaśnienie |
|---|---|
| MMTF<br>Mean Time<br>to Failure | Średni czas do awarii czyli średni czas poprawnego działania komponentu. |
| MTTR<br>Mean Time<br>to Repair | Średni czas naprawy czyli czas potrzebny do naprawy lub wymiany komponentu. |

Miary niezawodności komponentów sprzętowych

| Miara | Objaśnienie |
|---|---|
| POFOD<br>Probability<br>of failure<br>on demand | Prawdopodobieństwo awarii przy zleceniu. Prawdopodobieństwo, że system ulegnie awarii po zleceniu mu usługi. (POFOD=0.001 oznacza, że jedno zlecenie na tysiąc zakończy się awarią) |
| ROCOF<br>Rate of failure<br>occurence | Częstotliwość występowania awarii (intensywność awarii). Częstotliwość występowania nieoczekiwanych zachowań. (ROCOF=0.02 oznacza, że w ciągu 100 jednostek czasu zdarzą się dwie awarie) |
| MTTF<br>Mean time<br>to failure | Średni czas do awarii. Średni czas między zaobserwowanymi awariami systemu. (MTTF=500 oznacza, że co 500 jednostek czasu można oczekiwać jednej awarii) |
| AVAIL<br>Availability | Dostępność. Prawdopodobieństwo, że system jest dostępny dla użytkownika w określonej chwili. (AVAIL=0.998 oznacza, że na 1000 jednostek czasu system prawdopodobnie będzie dostępny w czasie 998 jednostek) |

Miary niezawodności oprogramowania

- Liczba awarii systemu w czasie obsługi ustalonej liczby żądań usług. Służy do wyznaczenia POFOD.
- Czas (lub liczba transakcji) między awariami systemu. Służy do wyznaczenia ROCOF i MTTF.
- Czas spędzony przy naprawie lub ponownym uruchomieniu systemu. Przy założeniu, że system musi być bez przerwy dostępny, ten pomiar umożliwia wyznaczenie AVAIL.

| Klasa awarii | Opis |
|---|---|
| Chwilowa | Zdarza się tylko dla niektórych danych wejściowych. |
| Trwała | Następuje dla wszystkich danych wejściowych. |
| Usuwalna | System może ją usunąć bez interwencji operatora. |
| Nieusuwalna | Usunięcie awarii wymaga interwencji operatora. |
| Nieniszcząca | Awaria nie powoduje zniszczenia stanu i danych systemu. |
| Niszcząca | Awaria powoduje zniszczenie stanu i danych systemu. |

Klasyfikacja awarii

Czynności prowadzące do opracowania specyfikacji niezawodności:

1. Dla każdego zidentyfikowanego podsystemu należy wskazać różne rodzaje możliwych awarii systemu i zanalizować konsekwencje tych awarii.
2. Na podstawie wyników analizy awarii systemu trzeba podzielić awarie na klasy.
3. Dla każdej rozpoznanej klasy awarii trzeba zdefiniować wymagania niezawodnościowe za pomocą odpowiedniej miary niezawodności.
4. Tam gdzie trzeba, należy określić niezawodnościowe wymagania funkcjonalne, w których wskaże się funkcjonalność systemu umożliwiającą zmniejszanie prawdopodobieństwa awarii krytycznych.

Przykład

Przykłady funkcjonalnych wymagań niezawodnościowych:

1. Należy wyspecyfikować zakres wartości, które mogą być wprowadzane przez operatora. System będzie sprawdzał, czy wszystkie dane wejściowe otrzymane od operatora mieszczą się w tym przedziale.
2. W procesie inicjowania system będzie sprawdzał, czy wszystkie dyski nie zawierają uszkodzonych bloków.
3. System musi być zaimplementowany za pomocą bezpiecznego podzbioru Ady i sprawdzony za pomocą analizy statycznej.

# Analiza zagrożeń i ryzyka

- **Rozpoznawanie zagrożeń**. Rozpoznaje się potencjalne zagrożenia, które mogą wystąpić. Ich zbiór zależy od środowiska, w którym system będzie użytkowany.
- **Analiza ryzyka i klasyfikacja zagrożeń.** Każde zagrożenie rozpatruje się oddzielnie. Te szczególnie poważne i niewykluczone są wybierane do dalszej analizy. W tym kroku można wyeliminować niektóre zagrożenia, ponieważ ich wystąpienie jest bardzo mało prawdopodobne. (np. jednoczesne uderzenie pioruna i trzęsienie ziemi)
- **Rozkładanie zagrożeń**. Każde zagrożenie rozpatruje się oddzielnie i szuka jego przyczyn.
- **Ocena szans zmniejszenia ryzyka**. Znajduje się propozycje sposobów zmniejszenia i redukcji rozpoznawanego ryzyka.

# Szacowanie ryzyka

- **Nie do przyjęcia**. System musi być zaprojektowany tak, że to zagrożenie nie może wystąpić, a nawet jeśli wystąpi, nie może doprowadzić do wypadku.
- **Tak małe, jak to możliwe** (ALARP — *as low as reasonable practical*). System musi być zaprojektowany tak, aby biorąc pod uwagę koszty, czas dostawy itd., zmniejszyć prawdopodobieństwo wystąpienia wypadku z powodu tego zagrożenia.
- **Akceptowalne**. Projektanci powinni przedsięwzięć wszystkie kroki w celu zmniejszenia prawdopodobieństwa powstania zagrożenia, jednak tylko pod warunkiem, że nie zwiększa to kosztu, nie opóźnia dostawy, ani nie pogarsza innych atrybutów niefunkcjonalnych.

Poziomy akceptowalności zagrożenia

# Specyfikowanie zabezpieczenia

- **Identyfikacja i wycena aktywów**. Rozpoznaje się aktywa (dane i programy) i ich oczekiwany stopień ochrony. (np. plik z hasłami ma zwykle większą wartość niż zbiór ogólnie dostępnych witryn WWW)
- **Analiza gróźb i oszacowanie ryzyka**. Rozpoznaje się możliwe groźby dla zabezpieczeń systemu i ocenia ryzyko związane z każdą z tych gróźb.
- **Przyporządkowanie gróźb**. Rozpoznane groźby przyporządkowuje się do aktywów.
- **Analiza technologii**. Ocenia się dostępne technologie zabezpieczeń i ich skuteczność na rozpoznane groźby.
- **Specyfikacja wymagań stawianych zabezpieczeniom**. Specyfikuje się wymagania stawiane zabezpieczeniom. Tam gdzie ma to sens, w specyfikacji wskazuje się technologie zabezpieczeń, które można zastosować w celu ochrony systemu przed groźbami.

# Wytwarzanie systemów krytycznych

- **Unikanie usterek**. W procesie projektowania i implementacji systemu należy przyjąć metody budowania oprogramowania, które umożliwiają minimalizację liczby błędów człowieka i wykrywanie usterek systemu przed przekazaniem go do użytku.
- **Tolerowanie usterek**. System należy zaprojektować tak, żeby usterki i niespodziewane zachowania działającego systemu były wykrywane i neutralizowane w taki sposób, aby nie doszło do awarii systemu.

# Unikanie usterek

- **Liczby zmiennopozycyjne** są ze swej natury niedokładne.
- **Wskaźniki** są konstrukcją niskiego poziomu. Przechowują adresy odwołujące się do obszarów pamięci maszyny. Są niebezpieczne, ponieważ błędy ich użycia powodują zniszczenia.
- **Dynamiczny przydział pamięci**. Pamięć programu może być przydzielana w czasie wykonywania, a nie w czasie kompilacji. Związane z tym niebezpieczeństwo polega na tym, że pamięć może nie być zwalniana, co w końcu powoduje całkowite wyczerpanie pamięci systemu.
- **Równoległość** jest niebezpieczna z powodu trudności z przewidywaniem subtelnych efektów czasowych zależności między procesami. Kłopotów z synchronizacją nie można wykryć przez kontrolę programów. Specyficzne zbiegi okoliczności, które powodują kłopoty z synchronizacją, mogą nie wystąpić w czasie testowania systemu.
- **Rekurencja** to sytuacja, w której procedura lub metoda wywołuje samą siebie albo inną procedurę, która z kolei wywołuje tę pierwszą. Jej zastosowanie może doprowadzić do zwięzłych programów. Śledzenie logiki programów rekurencyjnych jest jednak trudne. Błędy programistów są więc trudniejsze do wykrycia.
- **Przerwania** są sposobem wymuszenia przekazania sterowania do innego fragmentu kodu niezależnie od obecnie wykonywanego. Związane z nimi niebezpieczeństwa są oczywiste, ponieważ przerwanie może wstrzymać wykonywanie operacji krytycznej.
- **Dziedziczenie** w językach programowania obiektowego umożliwiają użycie wielokrotne i podział problemu, ale powoduje, że kod związany z obiektem nie znajduje się w jednym miejscu. To sprawia, że zrozumienie zachowania obiektu jest trudniejsze.

- **Synonimy** występują, gdy różne nazwy służą oznaczeniu tego samego bytu programu. Czytelnicy programu mogą łatwo przeoczyć instrukcje zmiany bytu, gdy muszą śledzić kilka jego nazw.
- **Domyślne przetwarzanie danych wejściowych**. Niektóre systemy mają domyślne ustawienia dotyczące przetwarzania danych wejściowych, niezależnie od danych przekazanych systemowi. To luka w zabezpieczeniu systemu, którą włamywacz może wykorzystać do przekazania programowi nieoczekiwanych danych wejściowych, których system nie odrzuci.

# Procesy tworzenia niezawodnego oprogramowania

- **Kontrola (inspekcja) wymagań** ma wykryć błędy w specyfikacji systemu. Duża część usterek w oprogramowaniu jest wynikiem błędów w wymaganiach.
- **Zarządzanie wymaganiami** polega na zapisywaniu zmian w wymaganiach i śledzeniu ich w trakcie projektowania i implementacji. Wiele błędów w dostarczonych systemach wynika z braku dbałości o uwzględnianie zmian wymagań w projekcie i implementacji systemu.

- **Weryfikacja modeli** polega na automatycznej analizie modeli systemu wykonywanej przez narzędzia CASE. Sprawdza się zarówno ich wewnętrzną, jak i zewnętrzną spójność. Spójność wewnętrzna oznacza spójność jednego modelu. Spójność zewnętrzna oznacza, że różne modele systemu (np. model stanu i model obiektowy) są spójne.

- **Kontrole projektu i kodu** są często realizowane na podstawie list kontrolnych z często występującymi usterkami. Ich celem jest wykrycie i usunięcie tych usterek przed przystąpieniem do testowania systemu.

- **Analiza statyczna** to zautomatyzowana metoda analizy programów, która polega na szczegółowym analizowaniu programu w poszukiwaniu błędnych fragmentów.

- **Planowanie i zarządzanie testowaniem**. Należy zaprojektować wyczerpujący zestaw testów systemu. Należy starannie zarządzać procesem testowania, aby upewnić się, że testy pokrywają wszystkie przypadki oraz że w pełni odpowiadają wymaganiom systemowym i projektowym.

# Tolerowanie usterek

- **Wykrywanie usterek**. System musi wykrywać wystąpienia szczególnych kombinacji stanów, które mogą prowadzić do awarii systemu.

- **Ocena szkód**. Należy wykrywać części systemu, na które awaria miała wpływ.

- **Odtwarzanie po usterkach**. System musi odtworzyć jakiś znany „bezpieczny” stan. Można to zrobić przez korektę stanu uszkodzonego (odtwarzanie po błędach do przodu) lub przez przywrócenie systemowi jakiegoś znanego „bezpiecznego” stanu (odtwarzanie po błędach wstecz).

- **Naprawa usterek**. Polega na takiej modyfikacji systemu, aby usterka nie powtórzyła się. W wielu wypadkach usterki oprogramowania ujawniają się w postaci stanów chwilowych. Ich przyczyną jest szczególna kombinacja danych wejściowych. Nie trzeba nie naprawiać, ponieważ natychmiast po odtworzeniu po usterce można wznowić normalne przetwarzanie. To istotnie odróżnia usterki w oprogramowaniu od usterek w sprzęcie.

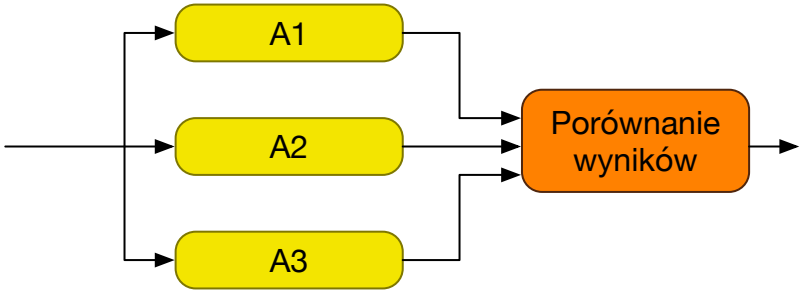
Cztery aspekty tolerowania usterek

- **Programowanie defensywne** to podejście do tworzenia programów, przy którym programiści zakładają, że w ich programach są niewykryte błędy i niespójności. Dodają oni nadmiarowy kod, który ma sprawdzać stan systemu po modyfikacjach i zapewniać, że zmiana stanu jest spójna. Po wykryciu niespójności anuluje się zmianę stanu lub przywraca jakiś znany poprawny stan.

- **Architektury tolerująca usterki** są architekturami systemów oprogramowania i sprzętu, które jawnie realizują tolerowanie usterek. Obejmują nadmiarowy sprzęt i oprogramowanie. Zawierają sterownik tolerowania usterek, którego zadaniem jest wykrywanie problemu i wspomaganie odtwarzania po usterkach.

Podejścia do implementacji tolerowania usterek

Przykład




Nadmiarowość trójmodułowa do radzenia sobie z awariami sprzętu