Programowanie funkcyjne

Informatyka I, 2 rok
Programowanie w języku Erlang

Pure functional

- Modelowanie funkcji matematycznych przy użyciu komputera
- Program to zbiór funkcji, których wartość można obliczyć dla dostarczonych danych
- Język funkcyjny
  - nie wykorzystuje zmiennych ani przypisań,
  - nie dostarcza pętli, skoków ani warunków.
- Program funkcyjny nie ma stanu
- Dla tych samych argumentów każda funkcja zawsze zwraca ten sam wynik.

Pure functional – zalety

- Brak skoków do innych miejsc programu
- Czytelność – matematyczny opis problemu
- Krótszy kod

Pure functional – krótszy kod?

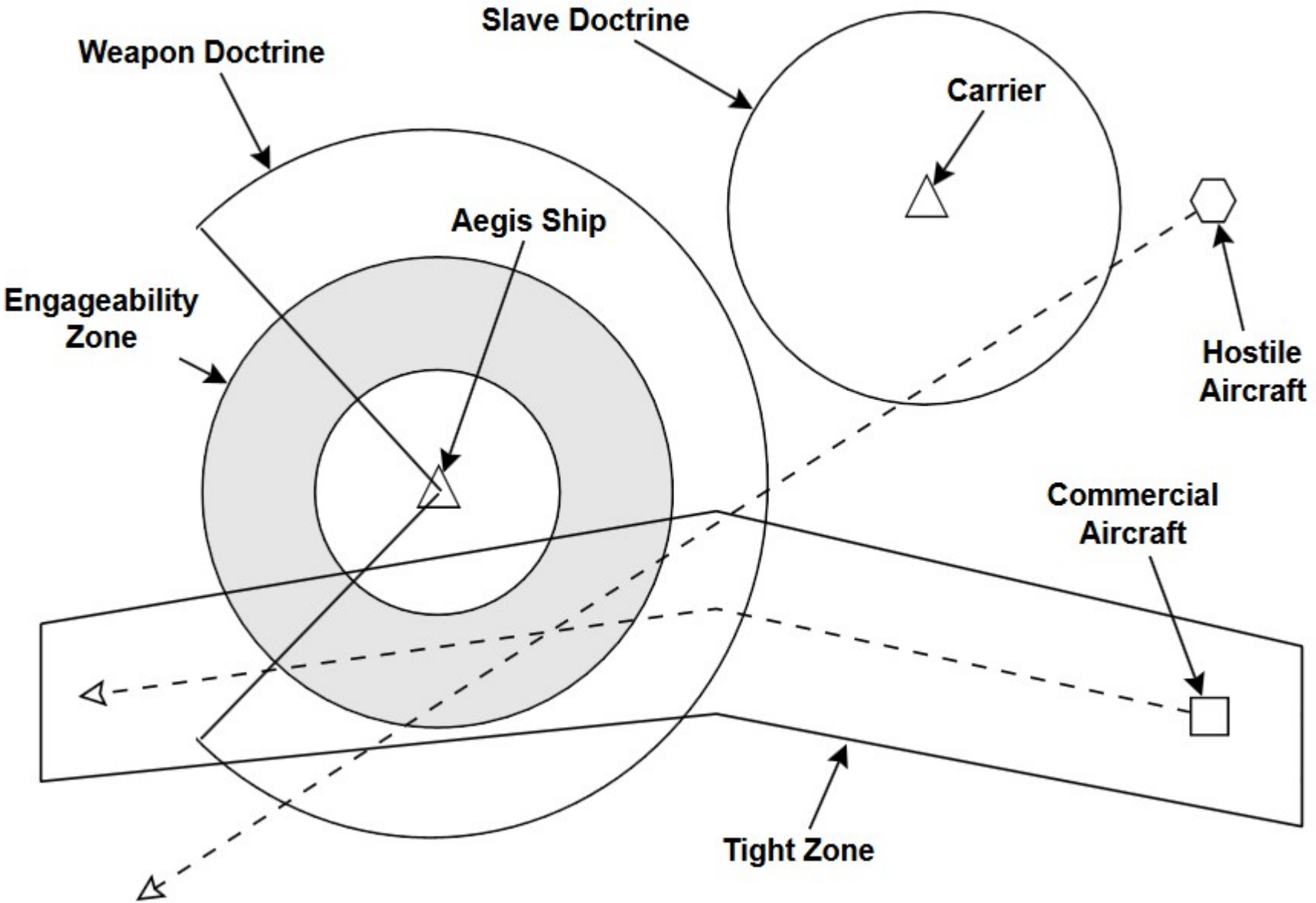
Haskell vs. Ada vs. C++ vs. Awk vs. ...
An Experiment in Software Prototyping Productivity*

Paul Hudak
Mark P. Jones

Yale University
Department of Computer Science
New Haven, CT 06518
{hudak-paul,jones-mark}@cs.yale.edu

July 4, 1994

Pure functional – krótszy kod?



Pure functional – krótszy kod?

Language	Lines of code	Lines of documentation	Development time (hours)
(1) Haskell	85	465	10
(2) Ada	767	714	23
(3) Ada9X	800	200	28
(4) C++	1105	130	–
(5) Awk / Nawk	250	150	–
(6) Rapide	157	0	54
(7) Griffin	251	0	34
(8) Proteus	293	79	26
(9) Relational Lisp	274	12	3
(10) Haskell	156	112	8

```
> type Region  =  Point -> Bool
```

Pure functional – zalety

- Brak skoków do innych miejsc programu
- Czytelność – matematyczny opis problemu
- Krótszy kod
- Testowanie
  - Działanie funkcji jest deterministyczne
  - Jeśli działa poprawnie, to zawsze działa poprawnie
- Współbieżność i równoległość
  - Wykonanie funkcji na różnych danych jest niezależne
  - Można to robić równolegle, bez synchronizacji

Podejście funkcyjne

- Pure functional nie działa – brak **efektów ubocznych**
- Komponowanie rozwiązania problemu z **funkcji**
  - Identyfikacja czynności ← funkcje
    - Dekomponowanie czynności ← funkcje
  - Identyfikacja danych ← funkcje?
- W języku funkcyjnym **funkcje są typem pierwszoklasowym** (First-class citizen)
  - Funkcje jako wynik działania programu
  - Funkcje jako argument innych funkcji

Podejście funkcyjne vs imperatywne

- Oblicz sumę liczb naturalnych mniejszych od 100 i podzielnych przez 3

```
int sum = 0;
for (int i = 1; i < 100; i++)
{
    if (i % 3 != 0)
      sum += i;
}
```

Jak to zrobić?

```
int sum = Enumerable.Range(0, 100).Where(i => i % 3 == 0).Sum();
```

```
List = [X  || X <- lists:seq(1,100), X rem 3 == 0].
Fun = fun(X, Sum) -> X + Sum end.
lists:foldl(Fun, 0, List).
```

Co zrobić?

To jest przyszłość!

- Popularne technologie obiektowe dodają rozszerzenia funkcyjne:
  - Java8,
  - C#,
- Języki oparte na podejściu funkcyjnym znajdują liczne zastosowania praktyczne:
  - Scala, Closure
  - F#
  - Erlang, Elixir
  - Ruby