Wstęp do Programowania potok funkcyjny

Marcin Kubica
kubica@mimuw.edu.pl

2018/2019

Outline

Trzy spojrzenia na informatykę

Trzy spojrzenia na informatykę

- Informatyka w szkole, OI i innych konkursach.
- Informatyka na uczelni.
- Informatyka w "przemyśle".

Informatyka w szkole, OI i innych konkursach.

Informatyka w szkole, OI i innych konkursach.

- Zawężony zakres:
  - problemy algorytmiczne,
  - wejście z pliku, wyjście do pliku,
  - liczy się wynik na testach.
- Zadany problem do rozwiązania.
- Dokładna specyfikacja.
- Jednoznacznie określony wynik.
- Można się domyślić oczekiwanej złożoności.
- Sukces: rozwiązanie przechodzi 100% testów.

Informatyka na uczelni

Informatyka na uczelni:

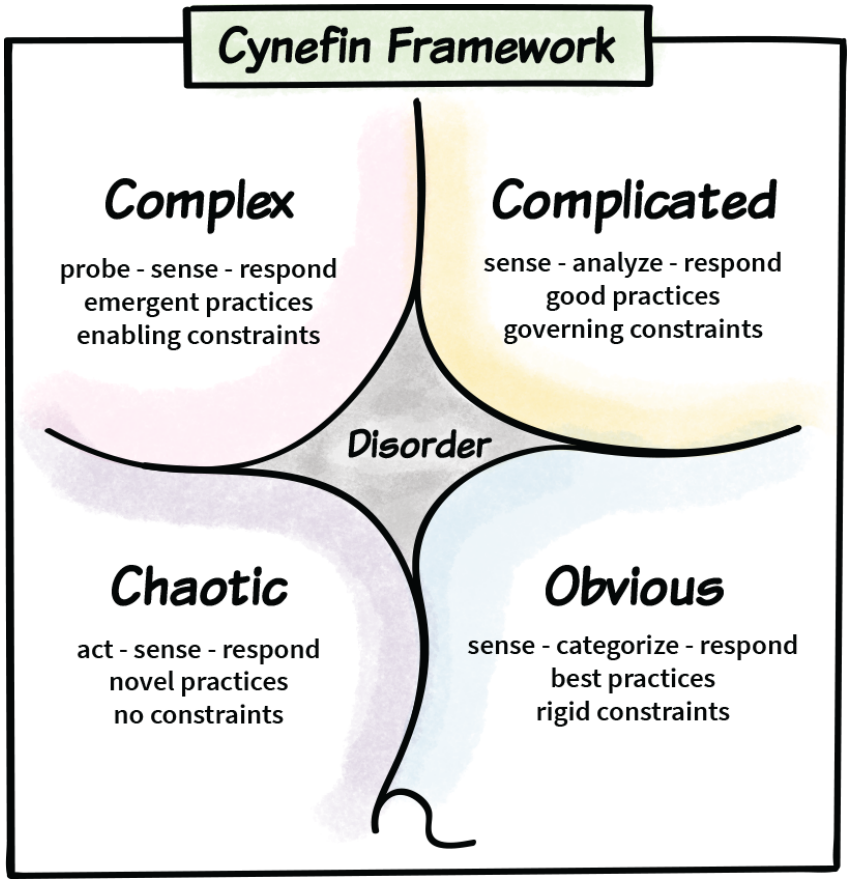
- Rozmaitość dziedzin i rodzajów problemów, nie tylko algorytmy.
- Ciekawy problem to połowa sukcesu.
- Problemy są (raczej) dobrze zdefiniowane.
- Nie wiadomo z góry jakie jest najlepsze rozwiązanie.
- Sukces: wynik lepszy od znanych dla „ciekawego” problemu.

Informatyka w przemyśle

Informatyka w przemyśle:

- Nie wiadomo jaki jest problem, co należy zaimplementować.
- Wszystkim się coś wydaje i wszyscy zmyślają.
- Użytkownicy też ...
- Wyzwanie: rozpoznać **potrzeby** użytkowników.
- A i tak nie wiadomo jakie skutki przyniesie implementacja.

Chaos, skomplikowane i złożone

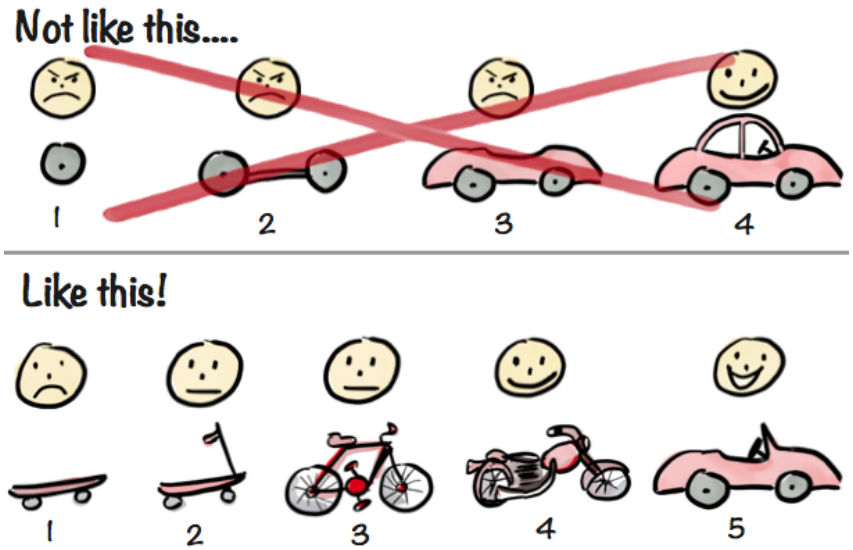


Problemy informatyczne są **ZŁOŻONE**.

Jak sobie radzić z problemami złożonymi

- Proces empiryczny:
  - zrób coś co wydaje Ci się najlepsze,
  - zbadaj skutki,
  - powtórz.
- Znajdź potrzebę do zaspokojenia, nie wymagania.
- Zbieraj feedback.
- Krótki feedback loop.

Zwinność



Scrum

Scrum

- Framework do pracy zespołowej.
- Do rozwiązywania złożonych problemów w sposób adaptacyjny.
- Najczęściej tworzenie oprogramowania, ale nie tylko.
- *Scrum Guide*

Scrum

Podstawy:

- Proces empiryczny.
- Filary: przejrzystość, inspekcja, adaptacja.
- Wartości:
  - zaangażowanie,
  - odwaga,
  - skupienie,
  - otwartość,
  - szacunek.

Scrum

Zespół:

- Product owner:
  - odpowiada za to Co? robić, a nie Jak?
  - maksymalizacja wartości wytworzonej przez zespół.
- Zespół deweloperski:
  - samoorganizujący się,
  - różnorodne kompetencje,
  - 5–9 deweloperów.
- Scrum master:
  - odpowiada za proces (sposób pracy),
  - pomaga wszystkim się polepszać.

Scrum

Rytuały i wydażenia:

- Sprint (iteracja)
  - ustalony czas 1-4 tygodni,
  - cel: dodanie (potencjalnie publikowalnej) wartości do produktu,
  - liczy się tylko to co skończone.
- Planowanie sprintu:
  - cel sprintu,
  - wyjaśnianie co jest potrzebne,
  - co zespół zobowiązuje się dostarczyć?
  - jak chcemy to osiągnąć?
  - czyszczenie backlogu.
  - 2 – 8 godz.

Scrum

Rytuały i wydażenia c.d.:

- Daily Scrum:
  - co udało się osiągnąć?
  - co planujemy osiągnąć?
  - co stoi nam na przeszkodzie?
  - timebox – 15 min. codziennie.
- Przegląd sprintu:
  - Zespół, PO (prowadzi), interesuariusze,
  - Jaki był cel, co się udało uzyskać, a co nie.
  - Prezentacja tego, co się udało uzyskać.
  - Feedback.
  - Przegląd backlogu, planów, sytuacji na rynku itp.
  - Pożywka dla PO do przygotowania backlogu przed następnym sprintem.
  - 1–4 godz.

Scrum

Rytuały i wydażenia c.d.:

- Retrospektywa:
  - „Grupowa sesja terapeutyczna”,
  - Refleksja nad tym jak się nam pracowało.
  - Co nam się podobało, co nie, co chcemy zmienić?
  - start / stop / kontynuacja / więcej / mniej
  - Po Review, a przed planningiem.
  - 1–3 godz.

Scrum

Artefakty:

- Product backlog
  - just-in-time planning
  - wieczna zmarzlina
  - miara postepu do celu.
- Sprint backlog
  - Podział na drobniejsze zadania implementacyjne.
  - Definition of Done.
- Increment.
- Przejrzystość i widoczność artefaktów.

Scrum i co dalej?

- Skalowanie: jak łączyć pracę wielu zespołów?
- Scrum nie zapewnia automatycznie zwinności, tylko ją umożliwia.
- Alternatywa: Kanban.
- Po co o tym opowiadam: EduScrum.

Kilka podstawowych pojęć

Algorytm vs. Program

Algorytm to sposób działania procesu obliczeniowego (niezależny od j. programowania)
Program to algorytm wyrażony w konkretnym j. programowania.

Specyfikacja Co można zakładać o danych?
Czego wymagamy od wyników?

Testowanie Szukanie kontr-przykładu, że program nie osiąga zamierzonego celu.

Weryfikacja Udowodnienie, że program osiąga zamierzony cel.

$$program = algorytm + język\ programowania$$

Dziedzina algorytmiczna

Definition

Dziedzina algorytmiczna, to zestaw elementarnych pojęć, których możemy używać opisując algorytmy:

- zbiory wartości,
- funkcje i stałe,
- relacje.

Dziedzina algorytmiczna jest „wbudowana” w język programowania. Możemy jednak mówić o niej niezależnie od języka programowania.

Dziedzina algorytmiczna, c.d.

Example

Dziedzina algorytmiczna starożytności:

- $\mathbb{R}_+$,
- 1, $+$, $-$, $\times$, $/$, $\sqrt{}$,
- $<$, $=$.

Jeśli mamy zbiór wartości logicznych { prawda, fałsz }, to relacje możemy utożsamić z funkcjami w zbiór wartości logicznych.

Zasady zaliczenia

- Kolokwia, terminy
- Programy zaliczeniowe
- Zwolnienia z egzaminu

Materiały

- Skrypt
  `http://www.mimuw.edu.pl/~kubica/wpf/`
- Moodle
  `https://moodle.mimuw.edu.pl/`
- Slajdy
  `http://smurf.mimuw.edu.pl/drupal6/?q=wstep_do_programowania_funkcyjny`
- Literatura:
  - Y. Minsky, A. Madhavapeddy, J. Hickey, *Real World OCaml*, O'Reilly 2014.
  - J. Tomasiewicz, *Zaprzyjaźnij się z algorytmami*, PWN 2017.
  - A. Hunt, D. Thomas, *Pragmatyczny programista*, WNT 2009.
  - H. Abelson, G. J. Sussman, *Struktura i interpretacja programów komputerowych*, WNT 2002.