Runtime software adaptation: approaches and a programming tool

mgr inż. Jarosław Rudy

Instytut Informatyki, Automatyki i Robotyki
Politechnika Wrocławska

11.09.2012

Plan prezentacji

1. Dziedzina Runtime software adaptation
2. Opis narzędzia
3. Badania
4. Podsumowanie

Runtime software adaptation - definicja

Runtime software adaptation

Ogół zmian w systemie informatycznym dokonywanych na etapie ewolucji (konserwacji) oprogramowania, przy czym zmiany te są wdrażane bez konieczności wyłączania i ponownego uruchamiania aplikacji.

Runtime software adaptation - definicja

Runtime software adaptation

Ogół zmian w systemie informatycznym dokonywanych na etapie ewolucji (konserwacji) oprogramowania, przy czym zmiany te są wdrażane bez konieczności wyłączania i ponownego uruchamiania aplikacji.

Cele adaptacji

- naprawa błędów programistycznych,
- dodanie nowych funkcjonalności,
- optymalizacja istniejących funkcjonalności.

Modele architektury oprogramowania

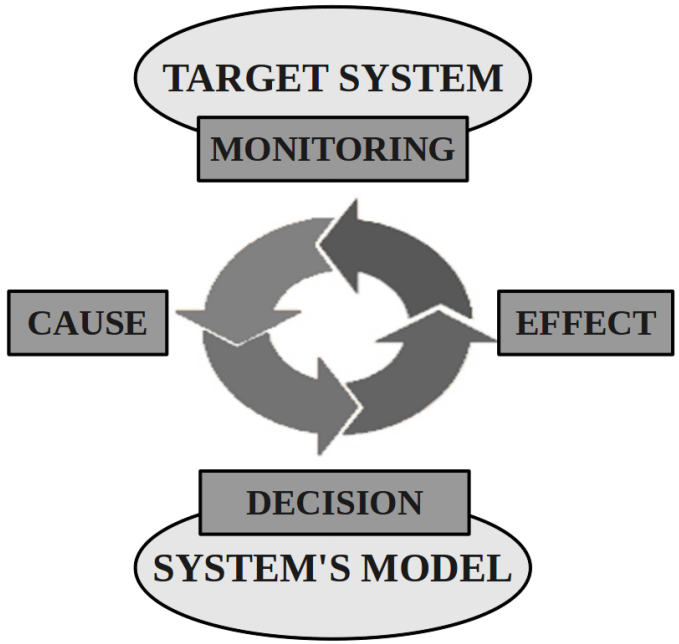
Idea

1. Tworzony jest model architektury systemu.
2. Tworzona jest implementacja systemu.
3. Zmiany modelu mogą zostać przekształcone na zmiany systemu.
4. Typową metodą reprezentacji modelu jest graf komponentów.

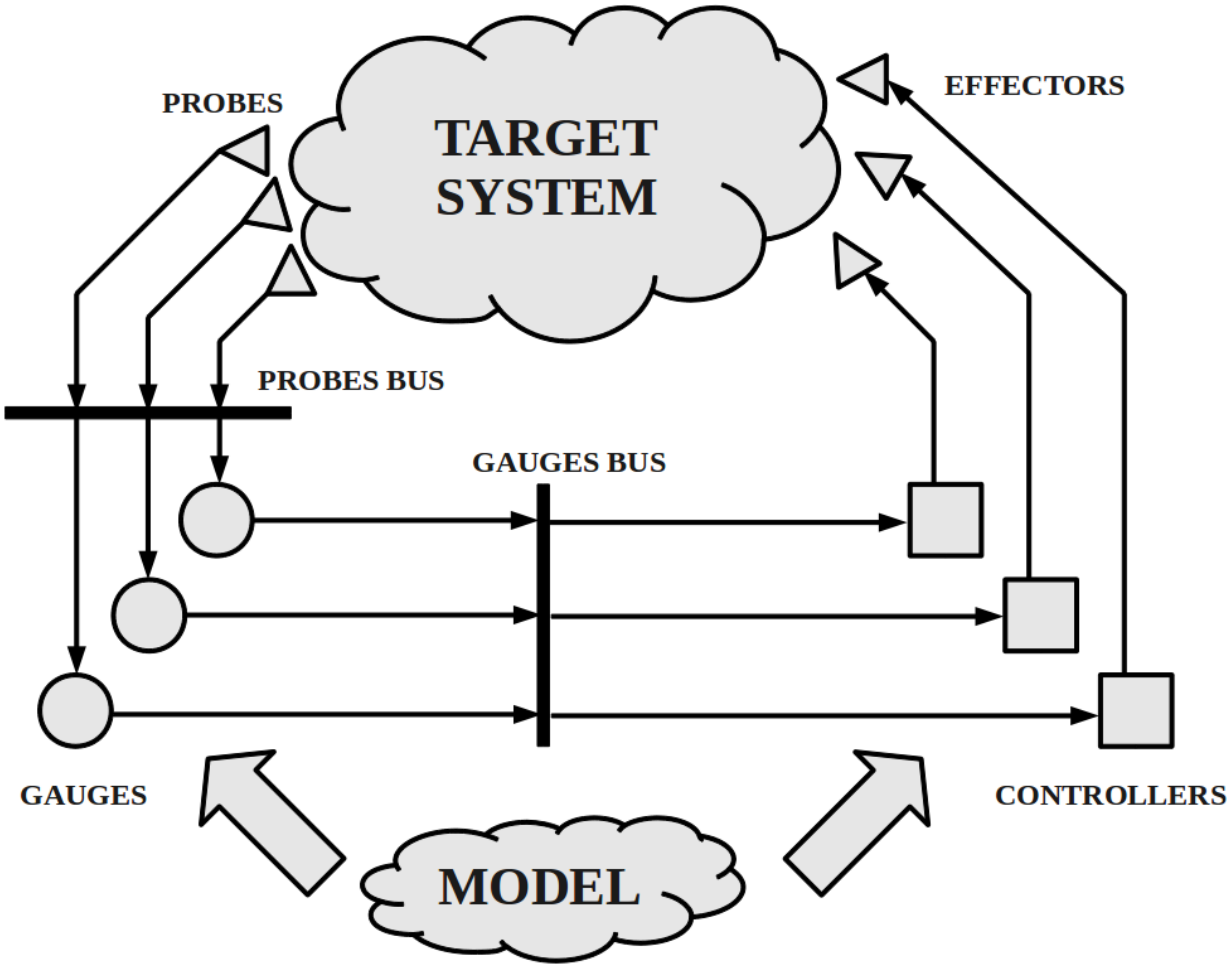
Modele architektury oprogramowania

Idea

1. Tworzony jest model architektury systemu.
2. Tworzona jest implementacja systemu.
3. Zmiany modelu mogą zostać przekształcone na zmiany systemu.
4. Typową metodą reprezentacji modelu jest graf komponentów.



Technika "próbników i oceniaczy"



Charakterystyka istniejących rozwiązań

- wymagana znajomość modelu architektury,
- konieczność projektowania oprogramowania zgodnie z regułami danego stylu architektury (nieefektywne w przypadku już istniejących systemów),
- postać rozbudowanych narzędzi (złożoność, trudność obsługi).

Versatile Code Generator - idea

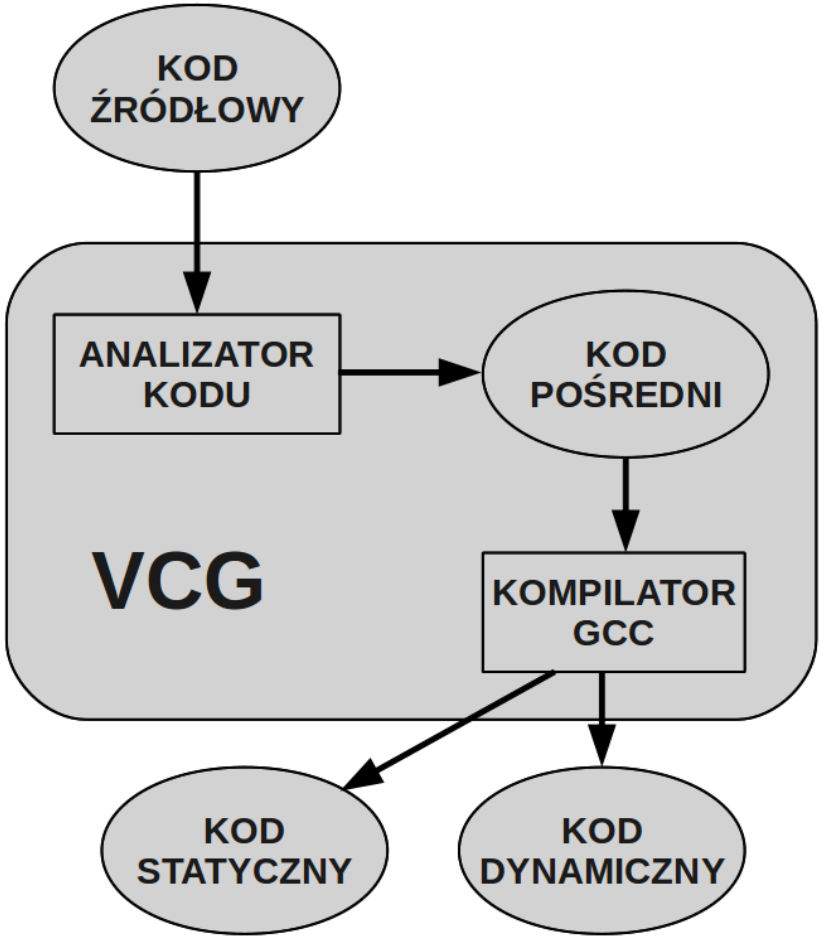
Idea

- narzędzie służące do tworzenia programów ze zdolnością dynamicznej podmiany kodu,
- działa na zasadzie preprocesora języka C/C++,
- wykorzystuje ładowanie bibliotek dynamicznych na żądanie,

Versatile Code Generator - idea

Idea

- narzędzie służące do tworzenia programów ze zdolnością dynamicznej podmiany kodu,
- działa na zasadzie preprocesora języka C/C++,
- wykorzystuje ładowanie bibliotek dynamicznych na żądanie,



Versatile Code Generator - funkcje dynamiczne

Funkcje dynamiczne

- poprzedzone dyrektywą `#versatile`,
- definicje umieszczone w osobnych plikach (bibliotekach dynamicznych),
- oryginalne wywołania zastąpione przez wywołania modułu zarządcy kodu.

Versatile Code Generator - funkcje dynamiczne

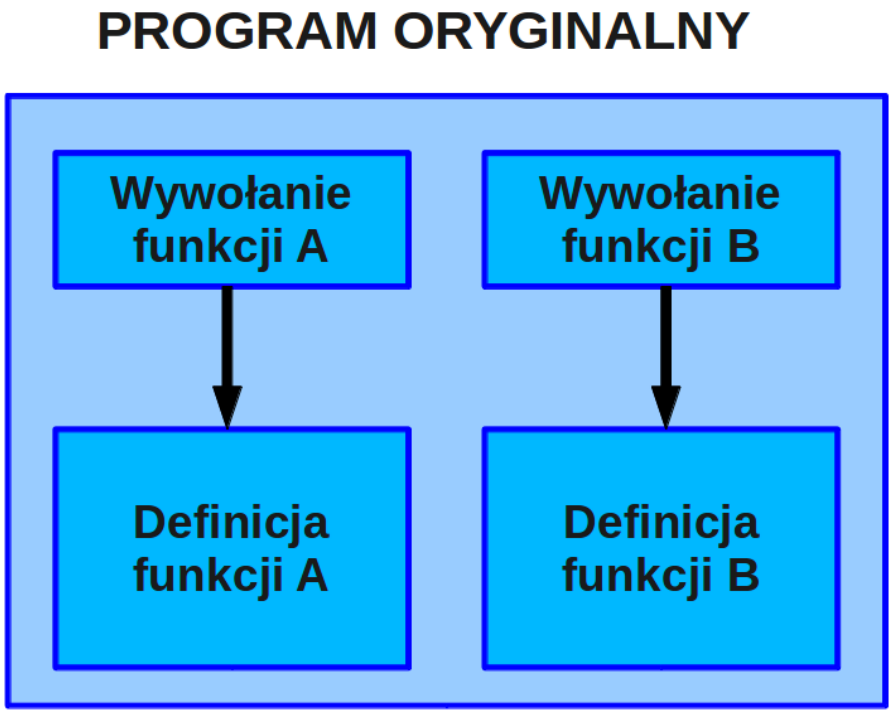
Funkcje dynamiczne

- poprzedzone dyrektywą `#versatile`,
- definicje umieszczone w osobnych plikach (bibliotekach dynamicznych),
- oryginalne wywołania zastąpione przez wywołania modułu zarządcy kodu.

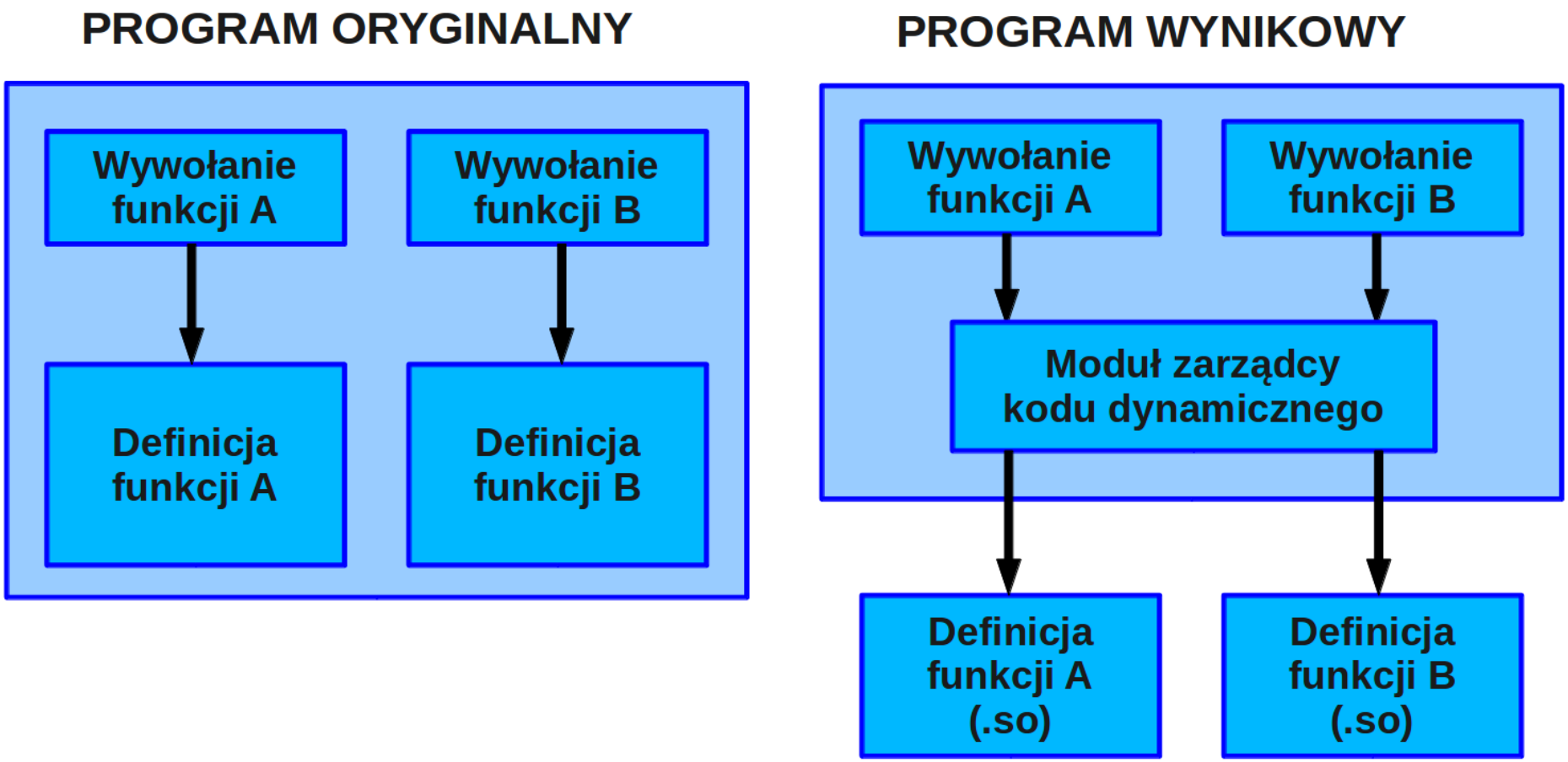
Zarządca kodu

- dodawany do każdego programu,
- wywołuje funkcje dynamiczne,
- odpowiedzialny za aktualizację zbioru funkcji dynamicznych.

Versatile Code Generator - architektura



Versatile Code Generator - architektura



Versatile Code Generator - obsługa błędów

Założenie pierwotne

- przechowywanie starszych wersji tej samej funkcji dynamicznej,
- w przypadku wystąpienia błędu (np. naruszenie ochrony pamięci) możliwość automatycznego powrotu do starszej wersji kodu bez konieczności wyłączenia aplikacji.

Versatile Code Generator - obsługa błędów

Założenie pierwotne

- przechowywanie starszych wersji tej samej funkcji dynamicznej,
- w przypadku wystąpienia błędu (np. naruszenie ochrony pamięci) możliwość automatycznego powrotu do starszej wersji kodu bez konieczności wyłączenia aplikacji.

W praktyce

- błąd powoduje wyłączenie aplikacji,
- starsze wersje funkcji również mogą być błędne.

Metodologia badań

Badania wydajnościowe

- czas wykonania,
- rozmiar kodu wynikowego,
- zapotrzebowanie na pamięć.

Metodologia badań

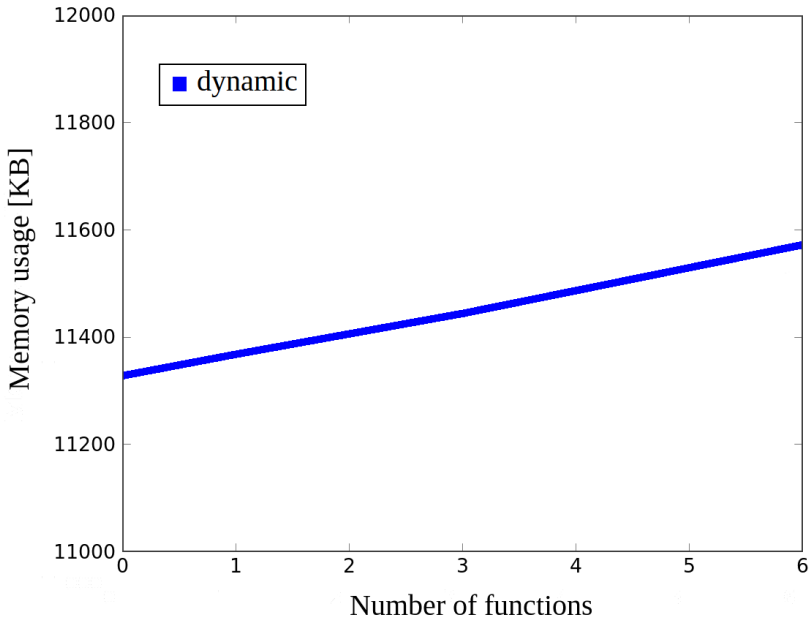
Badania wydajnościowe

- czas wykonania,
- rozmiar kodu wynikowego,
- zapotrzebowanie na pamięć.

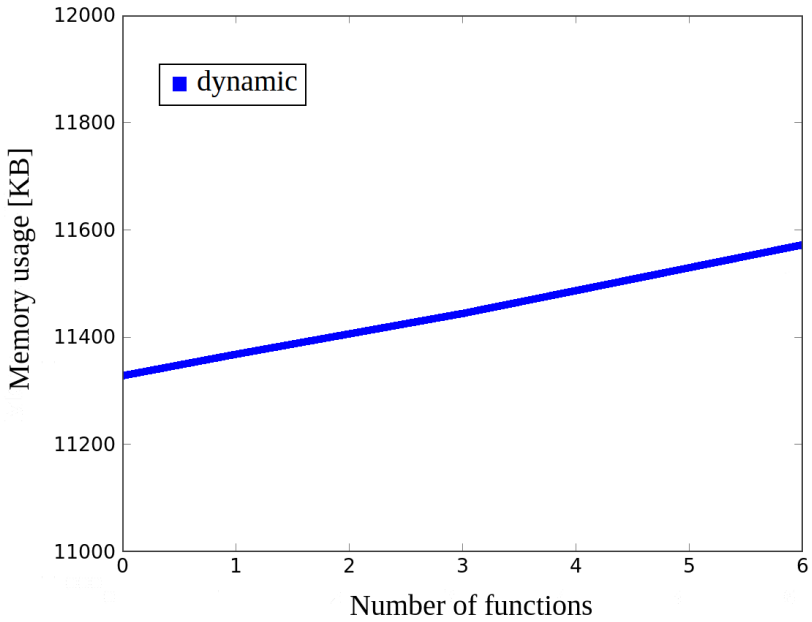
Badane zależności

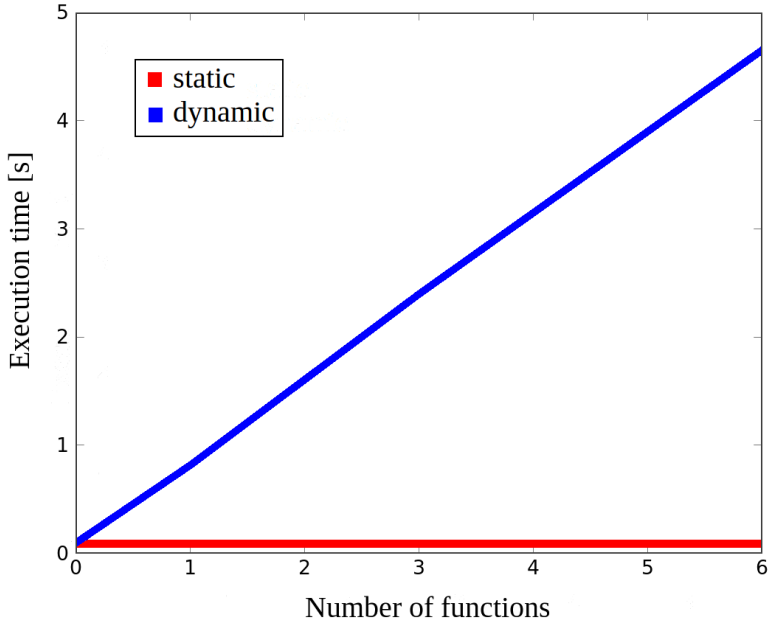
- liczba funkcji dynamicznych,
- rozmiar funkcji dynamicznych,
- narzut stały.

Badania - liczba funkcji

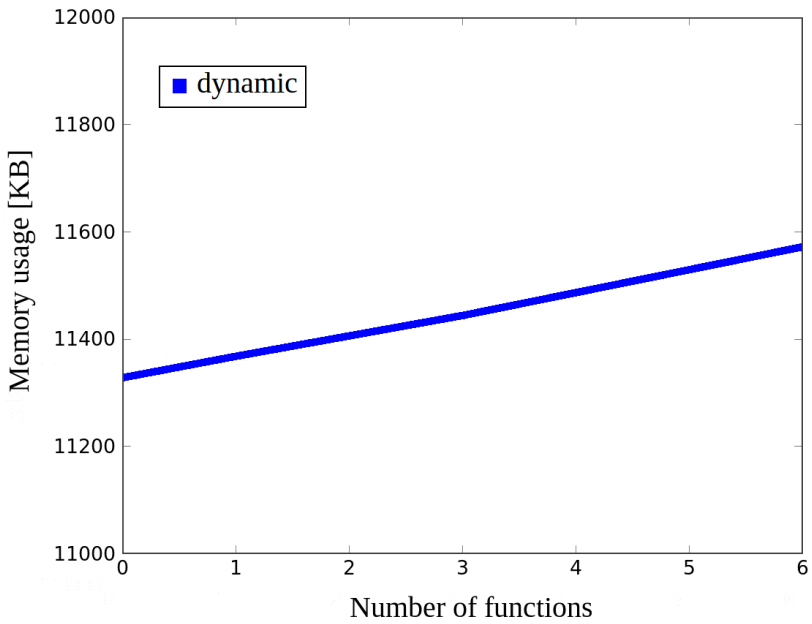


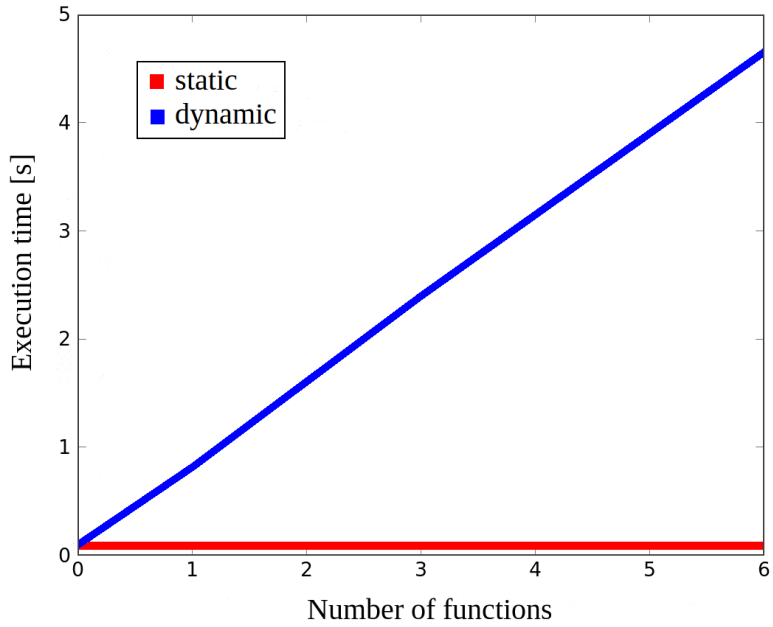
Badania - liczba funkcji





Badania - liczba funkcji





Rozmiar kodu wynikowego: $\sim$ 50 kB na każdą funkcję dynamiczną.

Badania - długość funkcji

Tablica: Rozmiar kodu wynikowego [kB]

Rozmiar funkcji	**bardzo mały**	**mały**	**średni**	**duży**
Statyczne	9.351	9.419	9.668	13.668
Dynamiczne	86.414	86.482	90.723	94.342
Różnica	**77.063**	**77.063**	**81.055**	**80.674**

Rozmiar funkcji

- bardzo mały - pojedyncza instrukcja `return`,
- duży - pętle, funkcje matematyczne oraz trygonometryczne.

Badania - narzut stały

Definicja

Różnica pomiędzy programem statycznym a dynamicznym bez deklarowania żadnych funkcji dynamicznych.

Tablica: Narzut stały

	Pamięć [kB]	**Kod wynikowy [kB]**	**Czas [ms]**
Statyczne	2960.0	9.327	2.5
Dynamiczne	11324.0	86.389	4.5
Różnica	**8364.0**	**77.062**	**2**

Podsumowanie - cz. 1

Cechy narzędzia

- tworzenie programów zdolnych do dynamicznej podmiany funkcji w trakcie działania i częściowej obsługi błędów,
- niskie zużycie zasobów, prostota użycia,
- minimalizacja udziału programisty,
- możliwość zastosowania w przypadku istniejącego oprogramowania,
- prototypowy charakter.

Podsumowanie - cz. 2

Analiza wydajności tworzonych programów

- umiarkowany wzrost kodu wynikowego,
- znaczny narzut stały na zużycie pamięci,
- znaczny wzrost czasu wykonania:
  - możliwa optymalizacja,

Literatura

A. Mukhija, M. Glinz.

Runtime Adaptation of Applications through Dynamic Recomposition of Components.
Proc. of 18th International Conference on Architecture of Computing Systems, 2005.

D. Garlan, B. Schmerl.

Model-based adaptation for self-healing systems.
Proceedings of the first workshop on Self-healing systems, WOSS '02, pp. 27–32, New York, NY, USA, 2002. ACM.

J. Dowling, V. Cahill.

The K-Component Architecture Meta-Model for Self-Adaptive Software.
In Akinori Yonezawa and Satoshi Matsuoka, editors, Proceedings of 3rd International Conference on Metalevel Architectures and Separation of Crosscutting Concerns (Reflection'2001), LNCS 2192, pp. 81–88. Springer-Verlag, 2001.

P. Oreizy, N. Medvidovic, R. N. Taylor.

Architecture-based runtime software evolution.
Proceedings of the 20th international conference on Software engineering, ICSE '98, pp. 177–186, Washington, DC, USA, 1998. IEEE Computer Society.

C. Parra, X. Blanc, A. Cleve, L. Duchien.

Unifying design and runtime software adaptation using aspect models.
Sci. Comput. Program., 76(12):1247–1260, December 2011.

G. Valetto, G. E. Kaiser, G. S. Kc.

A Mobile Agent Approach to Process-Based Dynamic Adaptation of Complex Software Systems.
Proceedings of the 8th European Workshop on Software Process Technology, EWSPT '01, pp. 102–116, London, UK, UK, 2001. Springer-Verlag.