UNIWERSYTET
MIKOŁAJA KOPERNIKA
W TORUNIU
Wydział Matematyki
i Informatyki

Prof. dr hab. Maciej M. Sysło
Wydział Matematyki i Instytut Informatyki
Uniwersytet Mikołaja Kopernika
ul. Chopina 12/18; 87-100 Toruń
tel. 0-604515777; syslo@mat.umk.pl; http://mmsyslo.pl

Toruń: 20.06.2018 r.

R e c e n z j a

rozprawy doktorskiej Pana **Kamila Żyły**

Inżynieria budowy aplikacji sterowanych modelami z dziedziny komunikacji personalnej

Celem pracy było zbudowanie graficznego języka modelowania, kompletnego w punktu widzenia dziedziny tworzenia aplikacji do komunikacji personalnej i umożliwiającego zwartą reprezentację jej problemów przy jednoczesnej łatwości użycia[1].

Zacznę od kwestii terminologicznej, która popsuła mi lekturę tej rozprawy. W kontekście swoich rozważań Autor wydziela dwie grupy osób: osoby o ograniczonej wiedzy technicznej i osoby techniczne[2]. Ta pierwsza grupa (str. 18), „nie jest zdolna (lub ma istotne trudności) do tworzenia oprogramowania z użyciem klasycznych metod/języków programowania. Trudności mogą wynikać z osobistych deficytów intelektualnych (problem z przyswojeniem/ wyuczalnością problematyki) lub z braku odpowiedniego wykształcenia.” Z kolei „osoba techniczna […] posiada umiejętności i doświadczenie wystarczające do samodzielnego tworzenia oprogramowania z użyciem klasycznych języków programowania.”

1. Jest NIEDOPUSZCZALNE i NAGANNE, by posądzać osobę, która nie „posiada umiejętności i doświadczenia wystarczającego do samodzielnego tworzenia oprogramowania, że może to być powodowane „osobistym deficytem intelektualnym”[3].
2. Propozycja podziału na osoby techniczne i nietechniczne w odniesieniu do umiejętności programowania nabiera szczególnego znaczenia obecnie, gdy do edukacji informatycznej w szkołach wprowadzono programowanie od pierwszej klasy szkoły podstawowej. Jednym ze środowisk programowania wizualno-blokowego w szkołach jest App Inventor, o którym jest mowa w p. 2.2.3. Podano tam liczby – 6 milionów użytkowników i 20 milionów aplikacji. Jeszcze więcej użytkowników regularnie odwiedza środowisko Scratcha i Godziny Kodowania (liczby odnośnie Polski można znaleźć na godzinakodowania.pl). Zgodnie z powyższym, to nie są osoby o ograniczonej wiedzy technicznej, gdyż programują, mam jednak wątpliwości, czy są to „osoby techniczne” w rozumieniu Autora[4]. A może po prostu NIE DZIELIC LUDZI!

---

[1] W wielu miejscach pracy Autor uzupełnia ten cel (tezę) o „jednoczesną akceptację ze strony osób o ograniczonej wiedzy technicznej”.

[2] W rozdz. 6 jest mowa o osobach związanych i niezwiązanych z kierunkami technicznymi.

[3] To jest przejaw dyskryminacji, który nie powinien pojawić się w pracy będącej podstawą awansu naukowego, oznaczającego również awans społeczny Autora. W innych społecznościach, na przykład amerykańskiej, byłoby to podstawą do relegowania ze społeczności. Niestety, Autor używa takiego uzasadnienia również swojej publikacji po angielsku [250].

[4] Warto pamiętać słowa znakomitego pedagoga Jeromego Brunera *any subject could be taught to any child at any age in some form that is honest* (J.S. Bruner, *The Process of Education*, Harvard University Press, 1960, 1977).

W świetle uwagi 2, określenie przez Autora grupy docelowej jest wewnętrznie sprzeczne – w jaki sposób „osoba **niezdolna** do tworzenia oprogramowania z użyciem klasycznych metod/języków programowania" ma posługiwać się zaproponowanym w pracy rozwiązaniem programistycznym?

Dalej, osobę z jednej lub z drugiej grupy nazywam użytkownikiem (*user*).

Organizacja pracy. Rozdział 1 zawiera definicje podstawowych pojęć i określenie celu badań. W rozdz. 2 dokonano przeglądu narzędzi użytecznych do tworzenia aplikacji do komunikacji personalnej. W rozdz. 3 zdefiniowano wymagania dotyczące tworzonego przez Autora graficznego języka modelowania AergiaML i przedstawiono jego koncepcję, a w rozdz. 4 zdefiniowano ten język. W rozdz. 5 opisano budowę środowiska wykonawczego dla tego języka w oparciu o platformę Eclipse. Rozdział 6 zawiera ewaluację postawionej tezy w oparciu o zestaw metryk wykorzystanych w badaniach. Konkluzje i podsumowanie zajęły rozdz. 7. W załącznikach umieszczono definicje metamodelu i elementy semantyki języka AergiaML

Rozdział 1. Podane tu określenia i definicje nie są zbyt precyzyjne. Dla przykładu, Komunikacja personalna jest określona jako komunikacja na poziomie człowiek-komputer i obejmuje czynności wykonywane przez użytkownika urządzenia osobistego[5] związane z wykorzystaniem modułów komunikacyjnych urządzenia oraz przechowywaniem danych. W wielu fragmentach pojawia się wymóg, by „graficzny język modelowania zapewniał kompleksową adresację pojęć dziedziny komunikacji personalnej" (str. 15 i inne). Co to jest „kompleksowa adresacja pojęć dziedziny komunikacji personalnej". Jaki mają z tym związek obszary wymienione na str. 17?

Rozdział 1 kończy sformułowanie tezy pracy, lista osiągnieć w pracy – komentuję ją w posumowaniu – oraz opis przebiegu badań z zaprojektowanym językiem.

Rozdział 2 zawiera przegląd i krytyczną analizę niektórych dotychczasowych rozwiązań – standardów, języków modelowania i narzędzi do ich wytwarzania – w zakresie modelowania zagadnień komunikacji personalnej. Autor uwzględnił blisko 25 rozwiązań, mógł jednak podarować sobie opis większości z nich, np. tych o których sam pisze, że zostały zarzucone albo mają wady, które je dyskwalifikują. Wiele z omówionych rozwiązań to propozycje z prac dyplomowych, które na ogół mają małą żywotność. W analizie nie posłużono się jakimiś formalnymi metodami, a więc krótkie opisy może stanowią o erudycji Autora, ale nic nie wnoszą do pracy, chociaż Autor mógł wynieść jakąś nauczkę z błędów w tych rozwiązaniach. W p. 2.3 przedstawiono dokładniejsze porównanie 10 języków modelowania, ale można mieć zastrzeżenia do pełności tego porównania, jeśli Autor pisze o braku dostępu do pełnej dokumentacji połowy z tych języków. Ponadto nie można porównywać rozwiązań aktywnych z tymi, które nie są rozwijane. Pozytywne oceny w tej analizie uzyskały UML i App Inventor, które dalej są porównywane z propozycją Autora – wystarczyło skupić uwagę tylko na tych językach.

Uwagi i ocena w dwóch ostatnich akapitach przed p. 2.4 jest zbyt ogólna, np. nie odnosi się do App Inventor jako środowiska dla użytkowników mniej doświadczonych w programowaniu. W p. 2.4 wystarczyło opisać EMP wykorzystywane dalej, a p. 2.4.5 jest kompletnie zbędny, jeśli cechy opisanych w nim narzędzi wykluczają ich użycie w pracy. Opisy standardów w p. 2.5 także wydają się po części zbędne – nie znalazłem w pracy odniesienia do nich.

Rozdział 3 Dyskusja i analiza w rozdz. 1-2 jest podsumowana w rozdz. 3, w którym Autor przedstawia swoją koncepcję graficznego języka modelowania AergiaML, wynikającą z krytycznej oceny stanu wiedzy przedstawionej w rozdz. 2. Podstawowe założenia projektowe zostały przedstawione w p. 1.2 a następnie rozwinięte w p. 3.1,1 i 3.1.2. Punkt 3.2 zawiera szczegółowe omówienie wymagań postawionych przed językiem AergiaML. Opis elementów

[5] Wśród urządzeń osobistych Autor wymienia tablety i smartfony, ale te urządzenia nie są w „równym stopniu" osobiste, np. tablet może być jedną z funkcji laptopa (np. w Lenovo Yoga klasy IBM PC).

języka AergiaML w tym rozdziale jest słowny, bez specjalnego wyróżnienia elementów tego opisu. Nie ma tutaj ścisłych powiazań między opisem w tym rozdziale a szczegółowymi wykazami w Załącznikach, zredagowanych w miarę formalnie. Ponadto, jest mowa o funkcjonalnościach zrealizowanych i tych, które tylko można zrealizować, oraz innych.

Wracam do użytkowników, którzy nie mają przygotowania technicznego. Na str. 70-71 Autor pisze: „użytkownik języka, który chciałby tworzyć aplikacje na własny użytek, poznawać wysokopoziomowe zagadnienia tworzenia aplikacji na urządzenia do komunikacji personalnej bądź uczestniczyć w procesie specyfikowania funkcjonalności oprogramowania […] Taki użytkownik musi być zdolny do odpowiedniego zrozumienia składni i wytworzonych z jej pomocą modeli, a także wydajnego jej wykorzystania.” Wyciąga stąd wniosek, że potrzebna jest składnia graficzna. Obawiam się, że taka składnia jest potrzebna nie tylko osobom bez przygotowania w zakresie programowania o czym świadczą przeprowadzone badania, które nie objęły jednak osób bez takiego przygotowania. Powyższe wnioskowanie nie jest więc uprawnione, co oczywiście nie wyklucza potrzeby istnienia graficznej składni – jest to obecnie powszechne podejście do tworzenia aplikacji dla użytkowników spoza grona specjalistów.

Następne **rozdziały 4-6** są opisem praktycznej realizacji nakreślonej koncepcji i jej ewaluacji – definicji języka i jego przykładowej implementacji. Wykorzystano w tym celu narzędzia udostępnione w Eclipse Modeling Project (EMP).

Rozdział 4 to opis definicji języka modelowania AergiaML. Autor korzysta tutaj z podejścia zaproponowanego przez Anneke Kleppe w [88], definiując składnię w oparciu o metamodel.

Uwaga do strony 78 i zapewne innej: co to jest „logika biznesowa”?

Omówiono tutaj dokładnie elementy składni języka podzielone na trzy grupy: komponenty. połączenia i kontenery. W p. 4.2 jest mowa o „dziedzinie komunikacji personalnej”, nie jest jednak jasne, na czym ma polegać „pokrycie” tej dziedziny, zwłaszcza kompletne, jak jest w wielu miejscach pracy[6]. Przedstawiono raczej wybrane elementy tej komunikacji, do kompletnego pokrycia to chyba jeszcze daleko. W p. 4.3 jest mowa o algorytmie przetwarzania modelu, który zilustrowano korzystając z serwisu TrashOut.

W **rozdziale 5** wykazano, że na bazie metamodelu języka można utworzyć środowisko wykonawcze zgodne ze standardami i interoperacyjne. Zilustrowano to przykładami. Użyto narzędzi EMP, zastosowano GMF.

Rozdział 6 jest poświęcony ewaluacji zaproponowanego języka. Badania faktycznie wykonano na grupie ok. 100 osób związanych z kierunkami technicznymi (z Instytutu Informatyki Politechniki Lubelskiej), które po szkoleniu przygotowawczym (20 godz.) miały wykonać modele trzech aplikacji: Dziadek, Gra i Pamiętnik w ciągu ok. 17 godz. z użyciem trzech języków: UML, App Inventor, AergiaML. W p. 6.2 zdefiniowano metryki dla modeli uzyskanych przez osoby biorące udział w eksperymencie, które posłużyły do ewaluacji tych modeli metodami statystycznymi. Osoby biorące udział w badaniach wypełniły również ankiety, które miały przynieść subiektywne oceny proponowanego języka. Badania miały na celu głównie ewaluację języka AergiaML na tle języków UML i App Inventor, szkoda jednak, że w omówieniu wyników nie poświecono więcej miejsca na porównanie przy okazji tych dwóch dodatkowych języków.

Jak sam Autor zauważył w odniesieniu do definicji niepoprawności (str. 153), nie uwzględniono w niej (i chyba nie tylko w niej) różnic konceptualnych pomiędzy technologiami, co okazało się niekorzystne dla AeriaML w przypadku niepoprawności modeli, zwłaszcza w po-

[6] Na str. 111 można przeczytać najpierw, że „Uzyskano znacząco lepsze pokrycie dziedziny niż w przypadku rozwiązań zidentyfikowanych w rozdziale 2”, jednak poniżej „można stwierdzić, że AergiaML adresuje dziedzinę w sposób kompletny/kompleksowy”. Brak precyzji i jednolitości w tym zakresie w całej pracy.

równaniu z App Inventor. Zapewne podobny argument, ale z odwróceniem ról można sformułować w odniesieniu do rozmiarów modeli.

W punkcie 6.3.5 autor wymienia słabe strony swojej propozycji, wśród nich połączenia. To przykład różnic konceptualnych w językach App Inventor i AeriaML. W tym pierwszym, połączenia są ściśle określone i kontrolowane w trakcie budowy modeli.

Uczestnicy badań z kierunków technicznych wysoko ocenili propozycje Autora w niemal wszystkich kategoriach.

W badanach wzięła również udział grupa osób nie związanych z kierunkami technicznymi. Jednak te osoby uczestniczyły tylko w kilkugodzinnych warsztatach, nie wykonały żądnego własnego modelu, wypełniły jedynie ankietę na temat proponowanego języka. Niestety tej części badań nie można uznać za dobrze przygotowaną i przeprowadzoną, dostarczającą miarodajnej oceny propozycji Autora. Co więcej, kładzie to cień na jeden z celów tej pracy – ewaluacji proponowanego języka przez osoby bez przygotowania technicznego. Należało dobrać osoby bez przygotowania technicznego (politechnicznego), ale zainteresowanych, w tym zawodowo, modelowaniem i tworzeniem aplikacji o wybitnie praktycznym znaczeniu. Wśród nich mogli się znaleźć np. studenci kierunków ścisłych, pracownicy ochrony czy instytucji ochrony środowiska. Przekrój międzynarodowy i zawodowy uczestników w tej grupie nie jest argumentem na korzyść tej części badań, bo nie dostarczyły one żadnych miarodajnych konkluzji.

Uwagi redakcyjne

Praca, chociaż starannie napisana i zredagowana, jednak miejscami jest przeładowana i „przegadana” – o czym piszę wcześniej – przez np. omawianie zagadnień, które mają niewielki związek z omawianą tematyką, lub są nieistotne (np. zostały zarzucone). Te dodatkowe materiały były zapewne źródłem inspiracji dla Autora, by nie podążać zarzucona drogą – chociaż nie wspomina o tym. W wielu miejscach brakuje precyzji sformułowań i redakcji, na co również wskazuję.

Autor nadużywa przypisów dolnych – jest ich 228 – np. nie wiadomo z jakiego powodu niektóre rozwinięcia akronimów znajdują się w tekście, a niektóre – w przypisach dolnych, a jest ich multum, nawet z przesadą, np. tłumaczenie, od czego pochodzą akronimy GPS, URL, WWW i SQL – to nadmierna drobiazgowość w pracy informatycznej. Wiele wyjaśnień u dołu powinno się znaleźć w głównym tekście, zapewne rozjaśniając go i skracając cały tekst, np. na str. 106

Językowo, pojawia się „szereg” jako liczebnik, a nim nie jest (np. „w szeregu przypadkach”, a wystarczy napisać „w wielu przypadkach”). Nadużywane jest „pozwala na”, gdy można użyć dokładniejszych kontekstowo sformułowań: „służy do”.

W p. 4.1 są opisane komponenty i kontenery, a dlaczego nie składowe i pojemniki? Autor wyjaśnia, że komponent dobrze brzmi, ale powołuje się na źródła obce [35] i [40].

Autor nadużywa cudzysłowów, np. w nazwach komponentów, wystarczy, że są pisane z wielkiej litery, to je wyróżnia (str. 87 i inne). Hugo Steinhaus nawoływał: „Nie cudzysłów!”.

Przyciski na ekranie, jak i klawisze na klawiaturze naciska się, a nie wciska – tego ostatniego fizycznie nie da się zrobić.

Polecam też: wyjątkowy, niepowtarzalny, jedyny w swoim rodzaju, zamiast „unikalny” – przymiotnik kojarzony z czasownikiem unikać.

Podsumowanie

Za istotny wkład w pracy należy uznać przeprowadzenie całego procesu zakończonego zaprojektowaniem, utworzeniem i ewaluacją „utworu” informatycznego – od specyfikacji po ewaluacje jego realizacji – jakim jest graficzny język modelowania AeriaML, spełniający w dużym stopniu tezę tej pracy. Wskazane przeze mnie usterki w pracy nie umniejszają

zbytnio tego osiągnięcia, a wskazywać mogą na dalsze kierunki doskonalenia warsztatu informatyka.

Biorąc pod uwagę wszystkie aspekty recenzowanej rozprawy w konkluzji stwierdzam, że rozprawa doktorska Pana **mgr inż. Kamila Żyły** pt. *Inżynieria budowy aplikacji sterowanych modelami z dziedziny komunikacji personalnej* spełnia wymogi Ustawy o stopniach naukowych i tytule naukowym i wnoszę o dopuszczenie Autora do dalszego toku przewodu doktorskiego.

Prof. dr hab. Maciej M. Sysło