Zalecenia odnośnie pisania prac inżynierskich

Krzysztof Podlaski

1 Uwagi ogólne

- Wszystkie prace muszą spełniać wymagania opisane w regulaminie prac dyplomowych.
- Część pisemna pracy inżynierskiej może być tworzona w dowolnej technologii (word, latex, openoffice, ...), proszę o przesyłanie do mnie wersji pdf. Osobiście polecam latex ale jest to tylko sugestia :-).
- W trakcie pracy nad treścią pracy nie poprawiam tekstu, jedynie dodaję komentarze do dokumentu pdf.
- Proszę o założenie na potrzeby projektu repozytorium git/bitbucket, abym miał dostęp do aktualnej wersji kodu programistycznego. Repozytorium może być publiczne lub prywatne, w przypadku prywatnego proszę o dodanie mnie do tego repozytorium.
- W trakcie pracy nad projektem inżynierskim proszę pamiętać o tym co nauczyliście się w trakcie przedmiotu inżynieria oprogramowania. Nie wymagam pracy w modelu kaskadowym (waterfall). Pewnie wygodniejsza jest jedna z metodyk iteracyjnych, zwinnych. Jednakże, każda z nich wymaga aby każda z poniższych faz występowała co najmniej jednokrotnie:
  - analizy wymagań,
  - projektowania,
  - implementacji,
  - testowania.

  W przypadku stosowania modelu iteracyjnego często fazy występują kilkukrotnie. Wymienione fazy muszą/powinny mieć swoje odzwierciedlenie w części pisemnej pracy inżynierskiej. Z mojego punktu widzenia

bardzo ważne są dwie pierwsze oraz ostatnia z faz, one odzwierciedlają profesjonalizm dyplomanta. Jakość pracy wykonanej w trakcie implementacji świadczy o kompetencjach technologicznych.

- Programistyczna praca inżynierska wymaga dobrej dokumentacji projektu (patrz Sekcja 3).
- Zalecam tworzenie bibliografii w miarę pisania pracy. Podejście "dodam później", niestety stosowane, zwykle wychodzi bokiem.
- To samo dotyczy formatowania, w przypadku korzystania z word bądź openoffice zalecam od samego początku wykorzystywać odpowiednie style, aby można było później łatwo zmieniać format całego dokumentu. Zostawianie tego na sam koniec powoduje często problemy i "rozjeżdżanie się" pracy.

2 Forma

2.1 Język

Praca powinna być napisana poprawnie w języku polskim. Proszę zwrócić uwagę na wszelakie anglicyzmy często występujące w slangu informatycznym. W miarę możliwości stosujmy polskie słowa, jeżeli nie da rady łatwo zastąpić słowa angielskiego, należy tak przeformułować zdanie aby nie była wymagana odmiana słowa, a co za tym idzie konieczności dodawania polskiej końcówki do słowa angielskiego.

Źle: Na potrzeby pracy usługi RESTowe zostały stworzone z wykorzystaniem frameworka Spring.

Poprawnie: Na potrzeby pracy w celu stworzenia usług typu REST wykorzystano framework Spring.

Dodatkowo sugeruję stosowanie twardych spacji w celu unikania wszelkiego rodzaju sierotek, przed ostatecznym wysłaniem pracy do APD proszę o sprawdzenie czy nie występują bękarty, sierotki, wdowy i szewcy [1].

2.2 Rozdziały

Nie ma ogólnych założeń co do podziału logicznego pracy, przykładowy podział pracy na rozdziały poświęcone kolejnym zagadnieniom:

1. Wstęp - powinien zawierać ogólny opis celów pracy

2. Analiza wymagań,
3. Opis metod i technologii wykorzystanych w pracy,
4. Opis implementacji (dokumentacja projektu),
5. Testy aplikacji,
6. Instrukcja instalacji/obsługi,
7. Podsumowanie,
8. Bibliografia.

2.3 Odwołania

2.3.1 Bibliograficzne

W pracy należy stosować jeden z trzech systemów cytowań (Vancouver System, Harvard System, Oxford System) opisanych szczegółowo w [2], regulamin dyplomowania zaleca wykorzystanie systemu Vancouver. W przypadku systemów Vancouver i Oxford bibliografia powinna być posortowana w kolejności pojawiania się odwołań w tekście, natomiast w systemie Harvard alfabetycznie wedle nazwisk pierwszych autorów.

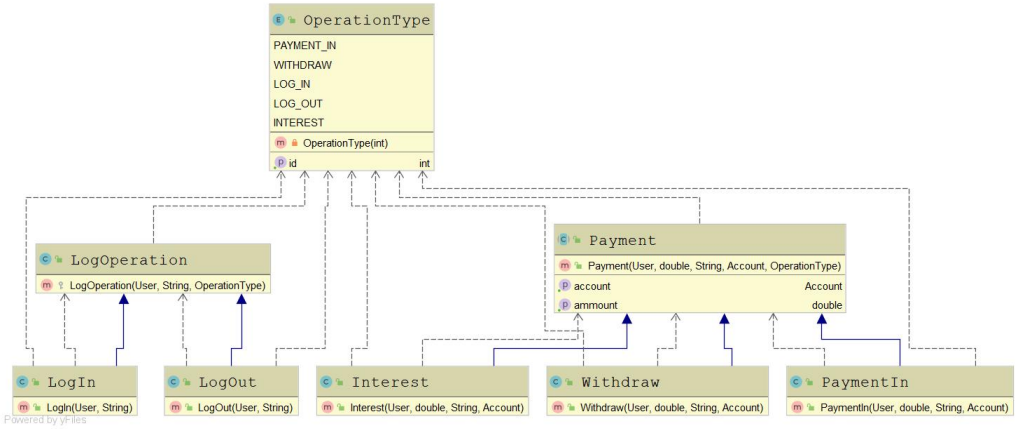
2.3.2 Elementy pływające

Każdy element pływający jak: obrazek, schemat, tabela czy fragment z kodem źródłowym powinien zostać odpowiednio oznaczony (ponumerowany) i zatytułowany. W pracy powinno znaleźć się co najmniej jedno odwołanie do wspomnianego elementu, w innym wypadku sprawia to wrażenie, iż element jest zbędny. Dodatkowo należy pamiętać, iż zasady umieszczania elementów pływających w tekście ograniczają znacząco jaką część strony mogą zawierać, a co za tym idzie omawiana tabelka, obrazek nie muszą znajdować się w bezpośrednim sąsiedztwie tekstu odnoszącego się do obiektu.

3 Dokumentacja Projektu

3.1 Wymagania aplikacji

Praca powinna zawierać analizę wymagań aplikacji, każda forma stosowana w inżynierii oprogramowania jest odpowiednia [3], dodatkowo do opisu założeń aplikacji należało by wykorzystać przypadki użycia (Use Cases) [4].



Rysunek 1: Przykładowy diagram klas w wybranym pakiecie i zależności pomiędzy nimi.

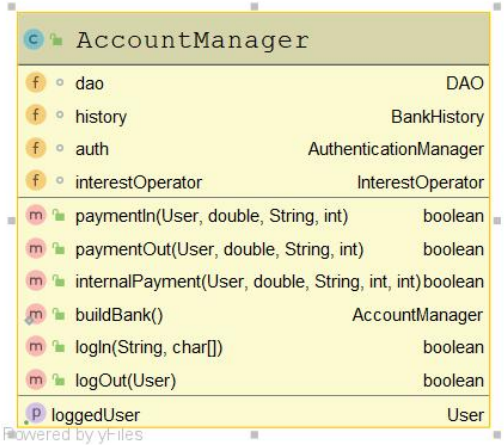
3.2 Architektura aplikacji/systemu

Bardzo często aplikacja/system tworzony w ramach pracy inżynierskiej składa się z wielu elementów jak np. baza danych, aplikacja serwerowa, aplikacja kliencka (mobilna). W pracy należy wyraźnie przedstawić podział na elementy składowe i metodę komunikacji pomiędzy nimi. Zalecanym sposobem przedstawienia zależności pomiędzy częściami projektu, oprócz opisu słownego, jest stosowanie schematów.

3.3 Dokumentacja kodu

Poprawnym sposobem dokumentacji struktury klas jest diagram klas. Możemy stosować różnego rodzaju diagramy, bardzo ogólny reprezentujący strukturę całej aplikacji Rys. 3, jak i diagramy opisujące poszczególne pakiety Rys. 1 bądź pojedyncze klasy Rys. 2. Do diagramów klas należy dołączyć bardziej szczegółowy opis klasy, informujący użytkownika za co klasa odpowiada, jakie są jej pola, metody i ich parametry, w tym celu można wykorzystać postać tabelaryczną (Tab. 1) lub przedstawić opis w tekście pracy. W przypadku prac zawierających znaczną ilość klas i ich metod nie wymagam aby wszystkie metody/pola zostały opisane w pracy. W części pisemnej należy opisać jedynie najistotniejsze metody/pola, dodatkowo należy stworzyć dokumentację techniczną klas za pomocą odpowiedniego narzędzia (doxygen, javadoc, ...) i dodać do pracy w postaci załącznika w formie elektronicznej (na płycie CD/DVD).

Działanie aplikacji, kodu aplikacji, powinno zostać zaprezentowane z wy-



Rysunek 2: Przykładowy diagram opisujący jedną klasę.

Nazwa metody	paymentOut
Opis metody	dokonanie przez użytkownika wpłaty na konto bankowe
parametry wejściowe	
user: User	reprezentuje użytkownika dokonującego wpłaty
ammount: double	kwota do wpłacenia
description : String	opis wpłaty
accountId: int	numer id konta na które należy dokonać wpłaty środków
parametry wyjściowe	
typ boolean	czy operacja zakończyła się sukcesem
Uwagi dodatkowe	
możliwe wyjątki	*OperationIsNotAllowedException* wyjątek tworzony w przypadku gdy użytkownik nie ma prawa wypłaty z podanego konta *SQLException* wyjątek tworzony w przypadku kiedy nie ma konta o zadanym accountid
podejmowane działania	W trakcie działania metody wszelkie operacje podlegają logowaniu z wykorzystaniem pola history

Tabela 1: Szczegółowy opis metody paymentOut klasy AccountManager.

korzystaniem diagramów UML [4]: sekwencji (Sequence diagram), aktywności (Activity diagram), stanu (State diagram) bądź schematów blokowych. Jeżeli autor pracy uzna, że fragment kodu jest niezbędny, należy zastosować krótki jego fragment (do 10-20 linijek kodu). Proszę o unikanie wklejania do dokumentu fragmentów kodu źródłowego aplikacji. Dodatkowo nie zalecam używania w tym celu zrzutów ekranu z środowiska programistycznego, powoduje to często różne rozmiary czcionek w zależności od rozmiaru wyciętego fragmentu. Dodatkowo kod powinien być kolorowany na białym tle, nie należy stosować motywów typu kolorowy tekst na czarnym tle - wygląda to dużo gorzej na papierze i zużywa mnóstwo tonera, tuszu. Mile widziana jest numeracja linii co ułatwia późniejsze odnoszenie się do przedstawionego kodu źródłowego. Kolorystyka i obramowanie kodu nie musi przypominać tego umieszczonego w dokumencie (Kod źr. 1), jednakże kod źródłowy musi się wyraźnie odróżniać od reszty tekstu pracy.

Kod źródłowy 1: Kod źródłowy obiektu app

```
1  /* Author: Krzysztof Podlaski, University of Lodz */
2  /* based partially on Apache Cordova default app */
3  var app = {
4      initialize: function() {
5          document.addEventListener('deviceready', this.
               onDeviceReady.bind(this), false);
6      },
7      onDeviceReady: function() {
8          this.receivedEvent('deviceready');
9      },
10     receivedEvent: function(id) {
11         var parentElement = document.getElementById(id);
12         var listeningElement = parentElement.querySelector('.
               listening');
13         var receivedElement = parentElement.querySelector('.
               received');
14         listeningElement.setAttribute('style', 'display:none;')
               ;
15         receivedElement.setAttribute('style', 'display:block;')
               ;
16         console.log('Received␣Event:␣' + id);
17     }
18 };
19 app.initialize();
```

	Pobranie wybranego rekordu
URL	`http://geniusgamedev.eu/cordova/rest_api/rest_srv/:id` lub `http://geniusgamedev.eu/cordova/rest_api/rest.php?id=:id`
Method	GET
Parameters	id - id wybranego elementu
Request Data	None
Response Data	obiekt w postaci: {'data':{ 'id':2, 'name':'Helena', 'score':128 } }

Tabela 2: Metoda get stosowanej usługi REST

	Dodanie nowego recordu
URL	`http://geniusgamedev.eu/cordova/rest_api/rest_srv` or `http://geniusgamedev.eu/cordova/rest_api/rest.php`
Method	POST
Parameters	brak
Request Data	nowy obiekt w postaci: { 'data': { 'id': 1, 'name': 'Jane', 'score': 332 } }
Response Data	obiekt w postaci: {'data':{ 'id':12, 'name':'Jane', 'score':332 } } nowa wartość id została przypisana przez serwer w trakcie rejestracji obiektu.

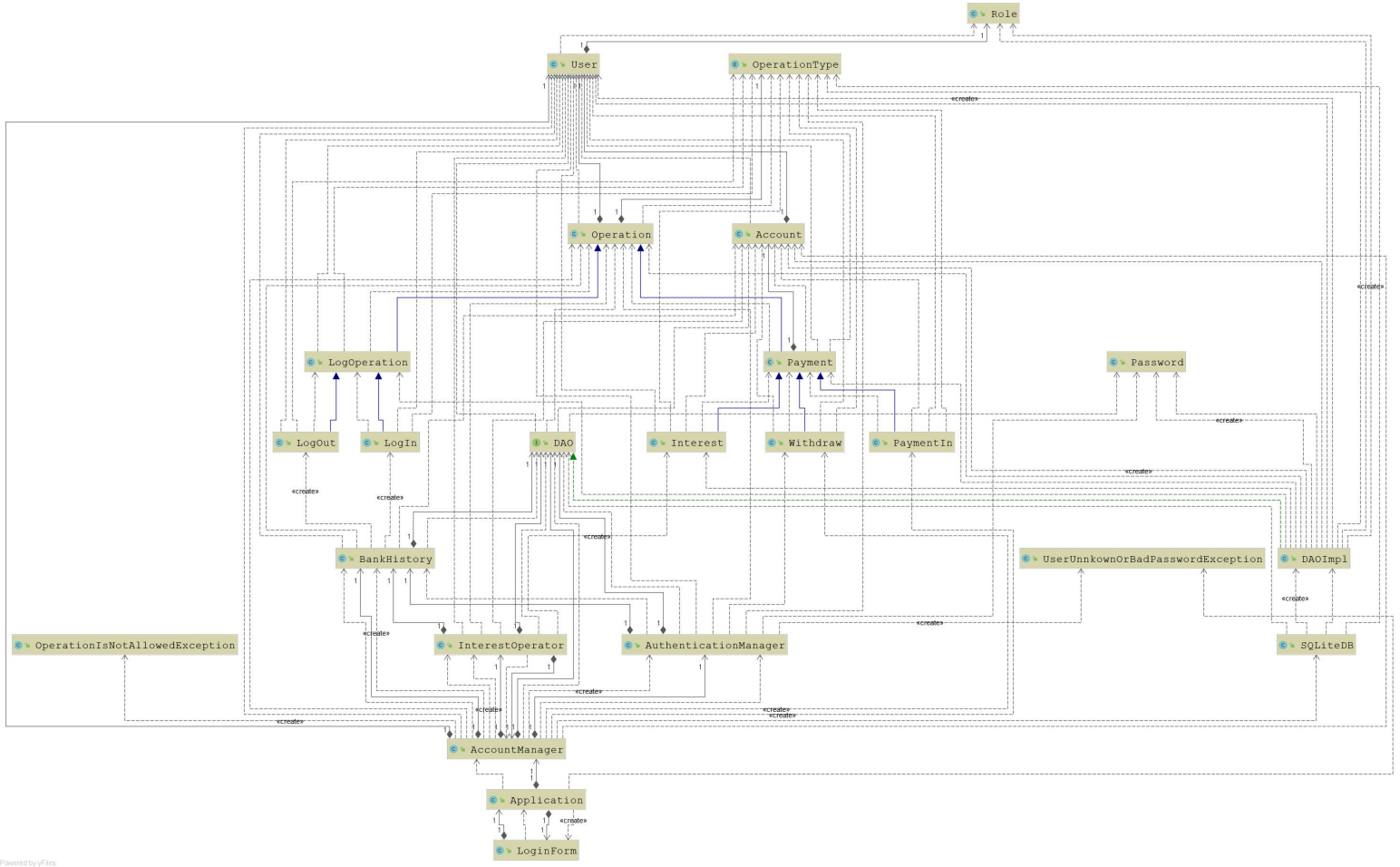
Tabela 3: Metoda post stosowanej usługi REST

3.4 Dokumentacja baz danych

Praca musi zawierać informację o strukturze wykorzystywanych baz danych. W przypadku stosowania baz relacyjnych zaleca się postać diagramów opisujących tabele i relacje pomiędzy nimi, w natomiast przypadku baz obiektowych diagramy obiektów.

3.5 Protokoły wymiany informacji

Jeżeli aplikacja zakłada wymianę danych pomiędzy elementami systemu, lub aplikacjami zewnętrznymi np z wykorzystaniem SOAP lub REST, należy opisać dokładnie protokół wymiany informacji, dla przykładu opis protokołu REST zawierają Tabele 2, 3.



Rysunek 3: Przykładowy ogólny diagram klas dla całej aplikacji.

3.6 Dokumentacja interfejsu użytkownika

Dokumentacja interfejsu użytkownika powinna pokazywać przepływ pomiędzy widokami, na przykład za pomocą diagramu stanów (State diagram), dodatkowo możliwe jest wykorzystanie schematów typu UI-mockups reprezentujących rozłożenie elementów widoku. Obrazy przedstawiające rzeczywisty wygląd ekranów aplikacji (zrzuty ekranów) powinny raczej zostać umieszczone w instrukcji obsługi.

3.7 Dokumentacja Testów

Na potrzeby pracy należy wykonać testy manualne i/lub automatyczne. Mile widziane jest zastosowanie testów jednostkowych, interfejsu użytkownika czy akceptacyjnych. Testy należy opisać w postaci: wyszczególnionych przypadków testowych, każdy przypadek testowy powinien zawierać opis sposobu wykonania, kroków niezbędnych do jego przeprowadzenia.

Literatura

[1] Strony Wikipedii poświęcone błędom w łamaniu tekstu
`https://pl.wikipedia.org/wiki/B%C4%99kart_(typografia)`

[2] Strona Wikipedii poświęcona metodom cytowania
`https://pl.wikipedia.org/wiki/Cytowanie_pi%C5%9Bmiennictwa`

[3] Ian Sommerville, Software Engineering, 10h ed., Pearsons (2016)
wydanie polskie Ian Sommerville, Inżynieria Oprogramowania, WNT (2003)

[4] Pascal Roques, UML in Practice: The Art of Modeling Software Systems Demonstrated through Worked Examples and Solutions, John Wiley & Sons, 2004
Przykładowy rozdział książki dotyczący wybranych diagramów UML i przypadków użycia `https://people.ok.ubc.ca/bowenhui/310/8-UML.pdf`

[5] A. De Vos, Reversible Computing: Fundamentals, Quantum Computing, and Applications. Weinheim: Wiley-VCH Verlag, Berlin (2010)

[6] M. Saeedi and I. L. Markov, "Synthesis and Optimization of Reversible Circuits: A Survey," ACM Computing Surveys, vol. 45, no. 2, pp. 21:1-34, 2013

[7] M. Soeken, R. Wille, O. Keszocze, D. M. Miller, and R. Drechsler, "Embedding of Large Boolean Functions for Reversible Logic," J. Emerg. Technol. Comput. Syst., vol. 12, no. 4, article 41, 26 pages, December 2015; also available as preprint arXiv.org:1408.3586, August 15, 2014

[8] C. Carlet, "Vectorial Boolean Functions for Cryptography," in: Y. Crama and P. Hammer (Eds,), Boolean Models and Methods in Mathematics, Computer Science, and Engineering, pp. 398-472 Cambridge University Press, 2010

[9] N. Tokareva, Bent Functions. Results and Applications to Cryptography," Academic Press, London 2015

[10] P. Kerntopf, C. Moraga, K. Podlaski, and R. Stankovic, "Towards Classification of Reversible Functions," In: Steinbach, B. (Ed.), Proceedings of the 12th International Workshop on Boolean Problems, September 2016, pp. 21-28

[11] P. Kerntopf, C. Moraga, K. Podlaski, and R. Stankovic, "Towards Classification of Reversible Functions with Homogeneous Component Functions," In: Steinbach, B. (Ed.), Further Improvements in the Boolean Domain. Newcastle upon Tyne: Cambridge Scholars Publishing, 2018, pp. 386-406

[12] P. Kerntopf, K. Podlaski, C. Moraga, and R. Stankovic, "Study of Reversible Ternary Functions with Homogeneous Component Functions," In: Proceedings of the 47th IEEE International Conference on Multiple-Valued Logic, May 2017, pp. 191-196

[13] P. Kerntopf, R. Stankovic, K. Podlaski, and C. Moraga, "Ternary/MV Reversible Functions with Component Functions from Different Equivalence Classes," In: Proceedings of the 48th IEEE International Conference on Multiple-Valued Logic, May 2018, pp. 109-114

[14] P. Kerntopf, K. Podlaski, C. Moraga, and R. Stankovic, "New Results on Reversible Boolean Functions Having Component Functions with Specified Properties," In: Soeken, M. (Ed.), Proceedings of the 13th International Workshop on Boolean Problems, September 2018, pp. 151-166

[15] C.-C. Tsai and M. Marek-Sadowska, "Boolean Functions Classification via Fixed Polarity Reed-Muller Forms," IEEE Transactions on Computers, vol. 46, no. 2, pp. 173-186, February 1997

[16] D. Debnath and T. Sasao, "Fast Boolean Matching under Variable Permutation Using Representative," Proceedings of the Asia and South Pacific Design Automation Conference, January 1999, pp. 359-362

[17] D. Debnath and T. Sasao, "Efficient Computation of Canonical Form for Boolean Matching in Large Libraries," Proceedings of the Asia and South Pacific Design Automation Conference, January 2004, pp. 591-596

[18] D. Debnath and T. Sasao, "Fast Boolean Matching under Permutation by Efficient Computation of Canonical Form," IEICE Transactions on Fundamentals of Electronics, vol. E87-A, December 2004, pp. 3134-3140

[19] D. Debnath and T. Sasao, "Efficient Computation of Canonical Form under Variable Permutation and Negation for Boolean Matching in Large Libraries," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E89-A, No.12, December 2006, Special Section on VLSI Design and CAD Algorithms, pp. 3443-3450

[20] R. S. Stanković, J. T. Astola, and B. Steinbach, "Former and Recent Work in Classification of Switching Functions," in: Steinbach, B. (Ed.), Proceedings of the 8th International Workshop on Boolean Problems, September 2008, pp. 115-126

[21] C. S. Lorens, "Invertible Boolean Functions," Space-General Corp., El Monte, CA, Re-search Memorandum No. 21; January 25, 1962 and July 1, 1962

[22] C. S. Lorens, "Invertible Boolean Functions," IEEE Transactions on Electronic Computers, vol. EC-13, no. 5, pp. 529-541, October 1964

[23] M. A. Harrison, "The Number of Classes of Invertible Boolean Functions," Journal of ACM, vol. 10, pp. 25-28, 1963

[24] I. E. Strazdins, "On the Number of Types of Invertible Binary Networks," Avtomatika & Vychislitelnaya Tekhnika, no. 1, pp. 30-34, 1974

[25] E. A. Primenko, "Invertible Boolean Functions and Fundamental Groups of Transformations of Algebras of Boolean Functions," Avtomatika & Vychislitelnaya Tekhnika, no. 3, pp. 17-21, 1976

[26] E. A. Primenko, "On the Number of Types of Invertible Boolean Functions," Avtomatika & Vychislitelnaya Tekhnika, no. 6, pp. 12-14, 1977

[27] E. A. Primenko, "On the Number of Types of Invertible Transformations in Multivalued Logic," Kibernetika, no. 5, pp. 27-29, 1977

[28] E. A. Primenko, "Equivalence Classes of Invertible Boolean Functions," Kibernetika, no. 6, pp. 1-5, 1984

[29] J. E. Rice, "Considerations for Determining a Classification Scheme for Reversible Boolean Functions," Technical Report TR-CSJR2-2007, University of Lethbridge, Department of Mathematics and Computer Science, Lethbridge, Alberta, Canada, 2007

[30] M. Soeken, N. Abdessaied, and G. De Micheli: "Enumeration of Reversible Functions and Its Application to Circuit Complexity," In: S. Devitt, I. Lanese (Eds.), Reversible Computation. Proceedings of the 8th International Conference, RC 2016, Bologna, Italy, July 7–8, 2016, Lecture Notes in Computer Science, vol. 9720, pp. 255-270, Springer International Publishing AG Switzerland, 2016

[31] T. G. Draper, "Nonlinear Complexity of Boolean Permutations" Ph.D. thesis, University of Maryland, 2009

[32] S. Aaronson, D. Grier, and L. Schaeffer, "The Classification of Reversible Bit Operations," Preprint arXiv:1504.05155 [quant-ph], 68 pages, April 20, 2015

[33] M. Caric and M. Zivkovic, "On the Number of Equivalence Classes of Invertible Boolean Functions under Action of Permutation of Variables on Domain and Range," Publications de l'Institut Mathématique, vol. 100, no. 114, 2016, pp. 95–99, also available as preprint arXiv:1603.04386v2 [math.CO], 9 pages, April 6, 2016

[34] J. Jegier, P. Kerntopf, and M. Szyprowski, "An Approach to Constructing Reversible Multi-Qubit Benchmarks with Provably Minimal Implementations," In: Proceedings of the 13th IEEE International Conference on Nanotechnology, pp. 99-104, 2013

[35] J. Jegier, P. Kerntopf, "Progress Towards Constructing Sequences of Benchmarks for Quantum Boolean Circuits Synthesis," In: Proceedings of the 14th IEEE International Conference on Nanotechnology, pp. 250-255, 2014