Łukasz Radliński

Ulepszone szacowanie ryzyka w przedsięwzięciach informatycznych z wykorzystaniem sieci Bayesa

Studia i Prace Wydziału Nauk Ekonomicznych i Zarządzania 20, 147-161

2010

STUDIA I PRACE WYDZIAŁU NAUK EKONOMICZNYCH I ZARZĄDZANIA NR 20

Łukasz Radliński

ULEPSZONE SZACOWANIE RYZYKA W PRZEDSIĘWZIĘCIACH INFORMATYCZNYCH Z WYKORZYSTANIEM SIECI BAYESA

Wprowadzenie

Wraz ze wzrostem wymagań tworzenia oprogramowania dobrej jakości przy niskich kosztach, rosną również wymagania w zakresie szacowania ryzyka w przedsięwzięciach informatycznych, rozumianych w tej pracy jako takie przedsięwzięcia (projekty), których celem jest wytworzenie oprogramowania. Pierwszym wymiarem szacowania w obszarze przedsięwzięć informatycznych jest szacowanie kluczowych czynników projektu. Istniejące modele skupiają się najczęściej na szacowaniu nakładów niezbędnych do realizacji projektu lub jakości wytworzonego oprogramowania. Chronologicznie pierwszymi modelami są modele parametryczne do szacowania liczby defektów[1], czasu między awariami[2] i nakładów[3].

[1] F. Akiyama, *An Example of Software System Debugging*, w: *Proc. Int. Federation for Information Processing Congress*, vol. 71, Ljubljana 1971, s. 353–379; M.H. Halstead, *Elements of Software Science*, Elsevier North-Holland, New York 1977.

[2] A.L. Goel, K. Okumoto, *An Analysis of Recurrent Software Failures in Real-Time Control System*, w: *Proc. ACM Annual Technology Conference*, Washington, D.C. 1978, s. 496–500; Z. Jelinski, P. Moranda, *Software Reliability Research*, w: *Statistical Computer Performance Evaluation*, ed. W. Freiberger, Academic Press, New York 1972, s. 465–484.

[3] B. Boehm, *Software Engineering Economics*, Prentice Hall 1981; L.H. Putnam, *A general empirical solution to the macro software sizing and estimating problem*, "IEEE Transactions on Software Engineering" 1978, vol. 4, no. 4, s. 345–61.

Jednym z najważniejszych zadań kierowników projektów jest odpowiednie balansowanie kluczowymi czynnikami projektu. Analiza kompromisu (ang. *trade-off*) pozwala szacować, jaka wartość jednego z tych czynników musi zostać poświęcona w celu osiągnięcia wyższej wartości innego czynnika. Na przykład: o ile większe nakłady są potrzebne w celu wytworzenia oprogramowania o określonej wielkości bez zmiany jakości procesu wytwórczego, ale o wyższym poziomie jakości oprogramowania? Albo: o ile mniejsza funkcjonalność oprogramowania zostanie dostarczona przy stałej jakości procesu, stałej docelowej jakości oprogramowania, lecz przy niższych nakładach? Z tego względu podjęto się napisania rozprawy doktorskiej pt. *Improved Software Project Risk Assessment Using Bayesian Nets*[4]. Praca ta skupiona była na analizie kompromisu między następującymi czynnikami projektu: nakładami, jakością procesu, dostarczoną funkcjonalnością (wielkością oprogramowania), jakością oprogramowania (liczbą defektów, wskaźnikiem defektów).

Drugim wymiarem szacowania w obszarze przedsięwzięć informatycznych jest szacowanie ryzyka. Popularne definicje określają ryzyko jako „źródło niebezpieczeństwa"[5] lub „możliwość poniesienia straty lub innej szkody"[6]. Pierwsza z tych definicji wskazuje na jakościowy wymiar ryzyka, natomiast druga na ilościowy. Podejście ilościowe często ujmuje ryzyko w postaci *wartości ryzyka* (inaczej: *podatność na ryzyko*):

$$\textit{wartość ryzyka} = \textit{prawdopodobieństwo zdarzenia} * \textit{wpływ zdarzenia} \quad \text{(równanie 1)}$$

Takie ujęcie ryzyka posiada jednak istotne wady: (1) szacowanie *prawdopodobieństwa zdarzenia* jest często trudne i pozbawione sensu bez analizy czynników wpływających na wystąpienie zdarzenia i działań mających na celu niedopuszczenie do wystąpienia zdarzenia, (2) szacowanie *wpływu zdarzenia* jest często trudne i pozbawione sensu bez analizy czynników i działań mogących zniwelować negatywne

[4] L. Radlinski, *Improved Software Project Risk Assessment Using Bayesian Nets*, Ph.D. Thesis, Queen Mary, University of London 2008.

[5] Webster's Online Dictionary, 2008, www.websters-online-dictionary.org, 1.06.2008.

[6] Merriam-Webster's Online Dictionary, 2008, www.merriam-webster.com, 1.06.2008.

konsekwencje, (3) *wartość ryzyka* nie posiada użytecznej interpretacji i może być wykorzystana jedynie do uszeregowania ryzyka[7].

Sieci Bayesa udostępniają mechanizm, który pokonuje sygnalizowane ograniczenia istniejących podejść w postaci jawnego ujęcia związków przyczynowo-skutkowych w modelu, a w konsekwencji dokładniej odzwierciedlającego rzeczywistość.

Praca składa się z jedenastu rozdziałów, spisu literatury oraz pięciu załączników. Rozdział 1 stanowi omówienie celu, hipotez i zawartości rozprawy. Rozdział 2 zawiera przegląd dotychczas zbudowanych modeli i wskazuje ich zalety i wady. W rozdziale 3 zaprezentowane zostały dane empiryczne, wykorzystane do budowy nowych oraz niektórych z istniejących sieci Bayesa. Rozdział 4 stanowi przegląd istniejących sieci Bayesa. W rozdziale 5 omówione zostały szerzej główne ograniczenia istniejących sieci Bayesa oraz rozwiązania niektórych problemów. Kolejne rozdziały poświęcone są realizacji głównych celów rozprawy – budowie nowych modeli prognostycznych: *model produktywności* (rozdział 6 i 7), *model wskaźników produktywności i defektów* (rozdział 8), *model typów defektów* (rozdział 9) oraz *uczący się model iteracyjnego testowania i naprawiania defektów* (rozdział 10). Rozdział 11 zawiera podsumowanie osiągniętych wyników.

Rozprawa została napisana pod kierunkiem prof. Normana Fentona (Queen Mary, University of London). Recenzentami rozprawy byli: prof. Mark Harman (King's College London), dr Tracy Hall (Brunel University) oraz, jako dodatkowy recenzent powołany w procesie nostryfikacji dyplomu, dr hab. Alberto Lozano Platonoff (Uniwersytet Szczeciński).

1. Cel i hipotezy pracy

Głównym celem pracy było stworzenie modeli prognostycznych, które będą w stanie udzielić informacji na następujące pytania:

1. Jakie zależności występują między funkcjonalnością, nakładami a defektami?
2. Jaki wpływ na funkcjonalność i jakość oprogramowania ma alokacja nakładów?
3. Jak zmiana jakości procesu wpływa na poziom funkcjonalności i jakości dostarczanego oprogramowania?

[7] N. Fenton, M. Neil, *Measuring your risks*, Agena White Paper, 2005, www.agenarisk.com, 1.06.2008.

4. Jeśli nie dysponujemy wymaganymi nakładami na realizację projektu, jaką część funkcjonalności musimy poświęcić, żeby dostarczyć oprogramowanie na zadanym poziomie jakości?
5. Jakiej funkcjonalności i jakości powinniśmy oczekiwać, jeśli bieżący projekt jest bardziej skomplikowany od poprzednich?
6. Jak zmiany niekontrolowanych cech projektu wpływają na szacowaną produktywność i jakość oprogramowania?
7. Jak uciążliwe będą defekty dla przyszłych użytkowników oprogramowania?
8. Ile defektów zostanie znalezionych i naprawionych podczas fazy testowania?

W pracy zostały zweryfikowane następujące hipotezy naukowe:

H1: Możliwe jest zbudowanie sieci Bayesa do szacowania ryzyka w przedsięwzięciach informatycznych, która pokonuje ograniczenia istniejących sieci Bayesa bez poświęcania ich podstawowych założeń. W szczególności taka sieć umożliwia przeprowadzanie analizy kompromisu między kluczowymi zmiennymi w przedsięwzięciu (nakłady, funkcjonalność, jakość produktu) i może być dostosowana do potrzeb konkretnej firmy informatycznej.

H2: Możliwe jest zbudowanie sieci Bayesa do szacowania produktywności zespołu programistycznego i wskaźnika defektów, która zawiera czynniki środowiskowe opisujące naturę wytwarzanego oprogramowania, czynniki opisujące proces wytwórczy zidentyfikowane podczas statystycznej analizy danych empirycznych oraz wiedzę ekspercką.

H3: Możliwe jest zbudowanie sieci Bayesa do szacowania proporcji różnych typów defektów skategoryzowanych według ich dotkliwości, która zawiera czynniki środowiskowe opisujące naturę wytwarzanego oprogramowania, czynniki opisujące proces wytwórczy zidentyfikowane podczas statystycznej analizy danych empirycznych oraz wiedzę ekspercką.

H4: Możliwe jest zbudowanie sieci Bayesa do prognozowania liczby defektów znalezionych i naprawionych w przyszłych iteracjach testowania i naprawiania defektów, która uczy się wartości swoich parametrów na podstawie obserwacji z przeszłych iteracji testowania i naprawiania.

2. Prezentacja metod i rozwiązań w badanym obszarze

Zbudowanych zostało wiele modeli do szacowania nakładów projektowych oraz jakości oprogramowania. Modele te zostały zbudowane z wykorzystaniem różnych technik statystycznych, sztucznej inteligencji czy uczenia maszynowego, takich jak: regresja wieloraka, dynamika systemowa, sztuczne sieci neuronowe, szacowanie przez analogię, zbiory przybliżone, logika rozmyta, podejścia Bayesowskie, wektory nośne, algorytmy genetyczne i inne. W rozprawie została dokonana szczegółowa analiza porównawcza różnych metod do szacowania nakładów i jakości (oraz pokrewnych zmiennych). Chociaż autorzy informują najczęściej o względnie dużej dokładności prognoz uzyskiwanej tymi modelami, istnieją poważne ograniczenia tych technik: dostępność danych empirycznych o minionych przedsięwzięciach, brak możliwości pełnego ujęcia zależności do analizy kompromisu projektowego, ograniczone ujęcie wiedzy eksperckiej oraz niepewności.

Krytycznym czynnikiem wyboru metody jest jak najmniejsza zależność od dostępnych danych przy jednoczesnej możliwości integracji z wiedzą ekspercką. Sieci Bayesa wydają się metodą z największym potencjałem do przeprowadzania różnego typu analiz szacowania ryzyka w przedsięwzięciach informatycznych, w szczególności analizy kompromisu najważniejszych czynników projektu. Do pozostałych zalet sieci Bayesa zaliczyć można: brak stałych osobnych list predyktorów i zmiennych zależnych, wnioskowanie wprzód i wstecz, jawne ujęcie niepewności, możliwość przeprowadzenia obliczeń przy niepełnych danych, możliwość łączenia danych jakościowych i ilościowych, intuicyjność modelu.

Sieć Bayesa jest probabilistycznym modelem złożonym z dwóch elementów:

1. Skierowany acykliczny graf zawierający zestaw węzłów sieci odzwierciedlających zmienne losowe (logiczne, nominalne, numeryczne dyskretne lub numeryczne ciągłe) oraz zestaw skierowanych powiązań łączących pary zmiennych.
2. Rozkłady prawdopodobieństw przypisane do węzłów: dla zmiennych niezależnych (bez „rodziców") – prawdopodobieństwa bezwarunkowe, dla zmiennych zależnych (posiadających co najmniej jednego „rodzica") – prawdopodobieństwa warunkowe.

W pracy doktorskiej zastosowano do obliczeń algorytm dekompozycji drzewiastej (ang. *junction tree*), który jest algorytmem wnioskowania dokładnego.

Algorytm ten został rozszerzony o dynamiczną dyskretyzację zmiennych numerycznych[8] zaimplementowaną w narzędziu AgenaRisk[9].

3. Charakterystyka problemu w literaturze przedmiotu

Prognozowanie nakładów projektowych i jakości oprogramowania jest celem wielu badań podejmowanych od początku lat siedemdziesiątych XX wieku. Najważniejszą wadą dotychczas opracowanych modeli jest brak poprawnego ujęcia zależności między wielkością projektu, nakładami projektowymi i jakością wytworzonego oprogramowania. Ta wada powoduje, że modele te mogą być wykorzystane tylko do mocno ograniczonych typów analiz, w szczególności do prognozowania jednej ściśle określonej zmiennej (na przykład jakości). Takie analizy dostarczają jednak znacznie ograniczonych informacji przydatnych do wspomagania podejmowania decyzji, a same modele wymagają kosztownego gromadzenia szczegółowych danych empirycznych.

Analiza kompromisu projektowego obejmuje w literaturze analizę zależności między różnymi zmiennymi, najczęściej między zakresem, czasem i kosztem projektu. W niektórych badaniach dodatkowo uwzględniana jest jakość wytworzonego produktu. W modelach zaproponowanych w tej pracy dokładnie analizowany jest wpływ czynników procesu na nakłady i jakość, natomiast nie jest uwzględniany koszt, który może być stosunkowo łatwo dodany do tych modeli i ich transformację z sieci Bayesa do postaci diagramów decyzyjnych.

Jedno z podejść prognozowania defektów oprogramowania w czasie trwającego projektu polega na dopasowaniu krzywej liczby defektów znalezionych w czasie testowania oprogramowania. Wiele z istniejących modeli ujmuje wielkość nakładów na testowanie (chociaż niektóre ich nie zawierają), jednak żaden nie ujmuje czynników jakości procesu. Istniejące modele zakładają, że jakość procesu wytwarzania oprogramowania (w tym testowania) jest niezmienna w czasie trwania projektu. Takie założenie nie zawsze jest prawidłowe, szczególnie w odniesieniu do długotrwałych projektów. W tej pracy zaproponowany został model, który zawiera czynniki jakości procesu, co pozytywnie wpływa na dokładność uzyskanych prognoz oraz jego możliwości analityczne.

[8] M. Neil, M. Tailor, D. Marquez, *Inference in hybrid Bayesian networks using dynamic discretization*, "Statistics and Computing" 2007, vol. 17, no. 3, s. 219–233.

[9] Agena, *AgenaRisk. Bayesian Network Software Tool*, 2008, www.agenarisk.com, 1.06.2008.

Analiza literatury oraz doświadczenia własne autora wykazały, że kierownicy projektu potrzebują modelu o większym potencjale analitycznym, który będzie mógł być łatwo skalibrowany i/lub rozbudowany oraz będzie ujmował niepewność i wiedzę ekspercką. Fenton i in.[10] jako pierwsi zaczęli budować realistyczne sieci Bayesa na potrzeby inżynierii oprogramowania. Niniejsza praca stanowi rozwinięcie tego podejścia przez dostarczenie nowych możliwości analitycznych.

4. Propozycje autorskich rozwiązań

W dysertacji zaproponowano zbudowanie czterech modeli: *modelu produktywności*, *modelu wskaźników produktywności i defektów*, *modelu typów defektów* oraz *uczącego się modelu iteracyjnego testowania i naprawiania defektów*.

Model produktywności

Głównym celem budowy *modelu produktywności* była możliwość przeprowadzania analizy kompromisu między kluczowymi czynnikami przedsięwzięć informatycznych. Istniejące modele MODIST: na poziomie projektu[11] i do prognozowania defektów[12] częściowo umożliwiały przeprowadzanie takich analiz i odniosły sukces w praktycznych zastosowaniach. Posiadały one jednak pewne ograniczenia, których wyeliminowanie lub ograniczenie stało się główną motywacją do zbudowania *modelu produktywności*[13]. Model ten posiada unikalne cechy, które nie były dostępne we wcześniejszych modelach:

[10] N.E. Fenton, M. Neil, *A Critique of Software Defect Prediction Models*, "IEEE Transactions on Software Engineering" 1999, vol. 25, no. 5, s. 675–689; N.E. Fenton, M. Neil, *Software Metrics: Roadmap*, w: *The Future of Software Engineering*. *Proc. 22nd Int. Conf. on Software Engineering*, ed. A. Finkelstein, ACM Press, 2000, s. 357–370.

[11] N.E. Fenton, W. Marsh, M. Neil, P. Cates, S. Forey, M. Tailor, *Making Resource Decisions for Software Projects*, w: *Proc. 26th Int. Conf. on Software Engineering*, IEEE Computer Society, Washington, D.C. 2004, s. 397–406.

[12] N.E. Fenton, M. Neil, W. Marsh, P. Krause, R. Mishra, *Predicting Software Defects in Varying Development Lifecycles using Bayesian Nets*, "Information and Software Technology" 2007, vol. 43, no. 1, s. 32–43.

[13] Ł. Radliński, N. Fenton, M. Neil, D. Marquez, *Improved Decision-Making for Software Managers Using Bayesian Networks*, w: *Proc. 11th IASTED Int. Conf. Software Engineering and Applications*, Cambridge, MA 2007, s. 13–19.

1. **Użycie wskaźników produktywności i defektów podanych *a priori* przez użytkownika na podstawie danych historycznych z minionych projektów.** Jeśli podanie tych wskaźników nie jest możliwe, wskaźniki te mogą być oszacowane przez dodatkowy *model wskaźników produktywności i defektów* omówiony w dalszej części.
2. **Przeprowadzenie analizy kompromisu między kluczowymi zmiennymi projektu wyrażonymi na skali numerycznej.** Wszystkie kluczowe zmienne: funkcjonalność, jakość oprogramowania i nakłady są wyrażone liczbowo, a nie na skali porządkowej, tak jak nakłady i jakość oprogramowania w niektórych istniejących modelach.
3. **Korzystanie z jednostek miar wybranych przez użytkownika.** Model umożliwia wyrażenie kluczowych zmiennych numerycznych w różnych jednostkach. Użytkownicy mogą wybrać jedną z predefiniowanych jednostek dla każdej zmiennej numerycznej, a nawet korzystać z jednostek wprowadzonych przez siebie.
4. **Wpływ czynników jakościowych może być łatwo dostosowany przez zmianę wag w definicjach poszczególnych zmiennych.** Dostarczony kwestionariusz może pomóc w poznaniu opinii użytkowników na temat zależności między zmiennymi modelu. Te opinie mogą być odmienne dla pracowników różnych firm i mogą zależeć od typu tworzonego oprogramowania, zastosowanych procesów wytwórczych i innych czynników. Odpowiednia kalibracja modelu zapewni precyzyjniejsze odzwierciedlenie rzeczywistości.
5. **Wprowadzenie docelowych wartości zmiennych numerycznych jako przedziałów wartości, nie tylko jako wartości punktowych.** Na przykład model jest w stanie odpowiedzieć na pytanie: jak możemy osiągnąć wskaźnik defektów w przedziale od 0,05–0,08 defektów na punkt funkcyjny w projekcie o określonej funkcjonalności i możliwych innych ograniczeniach?
6. **Zmienne numeryczne są dynamicznie dyskretyzowane.** Oznacza to, że definiowanie przedziałów dla rozkładu prawdopodobieństwa w zmiennych numerycznych następuje automatycznie w czasie przeliczania modelu. Nie tylko zwalnia to osoby budujące model z konieczności poświęcenia czasu na definiowanie tych przedziałów dla każdej zmiennej numerycznej, ale – co ważniejsze – zapewnia, że dyskretyzacja jest bardziej dokładna, gdyż bierze pod uwagę wszystkie obserwacje wprowadzone do modelu.

Model wskaźników produktywności i defektów

W praktyce wytwarzania oprogramowania występują sytuacje, kiedy nie jest możliwe oszacowanie wskaźników produktywności i defektów na podstawie danych z minionych projektów. Taka sytuacja ma miejsce, kiedy firmy nie gromadzą danych wymaganych do obliczenia tych wskaźników. Doświadczenie praktyczne autora oraz inne źródła[14] wskazują, że firmy często nie gromadzą danych o wykorzystanych nakładach. Bez danych o nakładach nie jest możliwe wyznaczenie wskaźnika produktywności. Czasami, nawet gdy te wskaźniki można obliczyć na podstawie danych historycznych, są one nieadekwatne do danej sytuacji. Dzieje się tak, gdy obecny projekt istotnie różni się od projektów realizowanych wcześniej pod względem architektury, użytych narzędzi czy też procesu jego wytworzenia. Zatem, konieczne jest rozwiązanie wymogu podania przez użytkownika wskaźników produktywności i defektów na podstawie najbardziej podobnych projektów z przeszłości.

Takim uzupełnieniem jest *model wskaźników produktywności i defektów* (MWPD). Celem tego modelu jest oszacowanie tych wskaźników na podstawie zmiennych opisujących naturę bieżącego projektu, a następnie przekazanie tych szacunków do *modelu produktywności*. Model ten posiada strukturę naiwnego klasyfikatora Bayesa. Powiązania pomiędzy zmiennymi niezależnymi a zależnymi nie odzwierciedlają związków przyczynowo-skutkowych. Taka struktura umożliwia jednak łatwiejszą budowę modelu, ponieważ zmienne są analizowane parami (jeden predyktor i jedna zmienna zależna). W efekcie predyktory nie są ze sobą powiązane.

Do budowy tego modelu został wykorzystany publicznie dostępny zbiór danych ISBSG[15] o minionych przedsięwzięciach informatycznych. Predyktory zostały zidentyfikowane na podstawie wyników analizy statystycznej (współczynnik korelacji rang Spearmana, test Kruskala-Wallisa, współczynniki Fi, V Cramera, kontyngencji, wykresy ramkowe i punktowe, dopasowanie rozkładów, histogramy). Wpływ zmiennych niezależnych na zmienne zależne odzwierciedla powiązania między poszczególnymi parami zmiennych, występujące w analizowanym

[14] A. Rainer, T. Hall, *Identifying the causes of poor progress in software projects*, w: *Proc. 10th Int. Symposium on Software Metrics*, Sep. 2004, s. 184–195.

[15] ISBSG, *Estimating, Benchmarking & Research Suite Release 9*, International Software Benchmarking Standards Group, 2005, www.isbsg.org, 1.08.2007.

zbiorze danych. Jednak te zależności zostały częściowo skorygowane innymi wynikami[16] oraz wiedzą wynikającą z doświadczenia praktycznego. Szczegóły dotyczące procesu budowy modelu i scenariusze ilustrujące sposoby korzystania z modelu zostały przedstawione we wcześniejszej pracy[17].

Model typów defektów

Większość modeli do prognozowania defektów traktuje wszystkie defekty jednakowo. Jednakże firmy potrzebują kategoryzacji defektów znalezionych w wytworzonych przez siebie produktach w celu zaplanowania kolejności naprawiania defektów. W pracy zastosowano trójstopniową kategoryzację defektów według ich szkodliwości (niewielkie, duże, ekstremalne) ze względu na dostępność danych do analizy w bazie ISBSG. Celem tego etapu badań było zbudowanie modelu do prognozowania proporcji poszczególnych typów defektów występujących w wytworzonym oprogramowaniu. *Model typów defektów* (MTD) powstał na bazie wyników analizy statystycznej przeprowadzonej według tej samej procedury, co w przypadku omówionego wcześniej MWPD.

Model ten jest również naiwnym klasyfikatorem Bayesa. *Proporcje defektów* to prognozowana zmienna typu nominalnego z trzema wartościami, których prawdopodobieństwo odzwierciedla proporcje defektów niewielkich, dużych i ekstremalnych. Taka struktura umożliwia łatwą kalibrację modelu dzięki zastosowaniu analogii prawdopodobieństwa do proporcji defektów – suma wynosi zawsze 1. Taka struktura posiada wadę polegającą na utracie rozkładu prawdopodobieństwa dla poszczególnych proporcji defektów – dostępna jest jedynie wartość punktowa.

MTD może być wykorzystywany do prognozowania jako niezależny model lub w połączeniu z *modelem produktywności*. W tym drugim przypadku dzięki połączeniu prognoz dotyczących proporcji defektów z MTD oraz prognozy całkowitej liczby defektów z *modelu produktywności* możliwe jest otrzymanie wyniku w postaci prognozowanych liczb defektów poszczególnych typów.

---

[16] C. Jones, Software Quality in 2002, *A Survey of the State of the Art*, Software Productivity Research, Inc., 2002, za: J.A. Sassenburg, *Design of a Methodology to Support Software Release Decisions (Do the numbers Really Matter?)*, Ph.D. Thesis, University of Groningen 2006.

[17] Ł. Radliński, N. Fenton, D. Marquez, *Estimating Productivity and Defects Rates Based on Environmental Factors*, w: *Information Systems Architecture and Technology: Models of the Organisation's Risk Management*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2008, s. 103–113.

Uczący się model iteracyjnego testowania i naprawiania defektów

Modele dyskutowane powyżej zakładają stałe i znane zależności między poszczególnymi zmiennymi. *Model iteracyjnego testowania i naprawiania defektów* (UMITND) jest modelem uczącym się, co oznacza, że model ten uczy się wpływu poszczególnych predyktorów na zmienne zależne z wykorzystaniem zbioru danych z przeszłości. Celem modelu jest prognozowanie liczby defektów znalezionych i naprawionych w procesie iteracyjnego testowania i naprawiania. Ten iteracyjny proces zakłada, że pełna funkcjonalność oprogramowania jest zaimplementowana przed rozpoczęciem fazy testowania. Testowanie i naprawianie defektów jest podzielone na serię iteracji trwających przez różny czas.

Model działa w następujący sposób: Użytkownik wprowadza dane opisujące proces minionych iteracji testowania i naprawiania oraz liczby defektów znalezionych i naprawionych w tych iteracjach. Model używa tych danych do oszacowania swoich parametrów: liczby *ukrytych defektów* oraz *mnożników* dla każdego czynnika procesu. Wartości tych mnożników odzwierciedlają wagę poszczególnych czynników procesu w kontekście ich wpływu na liczbę defektów znalezionych i naprawionych. Następnie użytkownik wprowadza planowane wartości czynników procesu w przyszłych iteracjach, na przykład odnośnie do planowanego poziomu nakładów przeznaczonych na testowanie i naprawianie. Do prognozowania liczby *znalezionych defektów* i liczby *naprawionych defektów* w przyszłych iteracjach model wykorzystuje te czynniki procesu oraz nauczone wcześniej *mnożniki* i liczbę *ukrytych defektów*. W tym momencie użytkownik może analizować różne scenariusze różniące się planowanymi nakładami, poziomem jakości procesu i ludzi i innymi czynnikami.

5. Ocena uzyskanych wyników

Przeprowadzona została wnikliwa walidacja zbudowanych modeli, jednak nie przy użyciu danych z rzeczywistych projektów z powodu braku dostępu do takich danych. Dostępny publicznie zbiór ISBSG nie zawiera czynników jakości procesu wykorzystanych w modelach. Inny zbiór[18] zawiera takie czynniki, ale brakuje w nim

[18] N. Fenton, M. Neil, W. Marsh, P. Hearty, Ł. Radliński, P. Krause, *On the effectiveness of early life cycle defect prediction with Bayesian Nets*, "Empirical Software Engineering" 2008, vol. 13, no. 5, s. 499–537.

danych o alokacji nakładów oraz czynników opisujących projekty. Obszerny przegląd publicznie dostępnych zbiorów danych[19] wykazał, że nie zostały opublikowane pełne dane wymagane do modeli zaproponowanych w pracy doktorskiej.

Ponadto, *model produktywności* przed użyciem wymaga kalibracji do potrzeb konkretnej firmy. Taka kalibracja może być przeprowadzona właściwie jedynie w odniesieniu do pojedynczej firmy, co zapewniłoby poprawne odzwierciedlenie w modelu wiedzy eksperckiej. Zatem, walidacja dokładności prognoz modelu z użyciem rzeczywistych danych wymagałaby użycia danych z tej konkretnej firmy.

Z powyższych powodów przeprowadzona została wewnętrzna walidacja modeli pod kątem poprawności ujęcia wiedzy eksperckiej oraz wyników dodatkowych analiz statystycznych, w szczególności kierunku zależności między poszczególnymi zmiennymi. Dokładność prognoz *modelu produktywności* została zweryfikowana przez porównanie tych prognoz z danymi pozyskanymi w badaniu ankietowym.

Analiza różnych scenariuszy w zbudowanych modelach wykazała duży potencjał tych modeli przy szacowaniu ryzyka projektowego. Zbudowane modele, szczególnie MWPD oraz MTD, które są naiwnymi klasyfikatorami Bayesa, mogą być stosunkowo łatwo rozbudowane o nowe zmienne, na przykład miary kodu programu. Jednak dodanie takich miar do modelu powoduje, że będzie on mógł być skutecznie wykorzystany dopiero na takim etapie tworzenia oprogramowania, kiedy wartości tych miar będą dostępne.

Działanie UMITND zostało zweryfikowane przy użyciu zbioru danych wygenerowanego w sposób częściowo losowy dla hipotetycznego projektu. Te dane zostały w procesie testowania modelu potraktowane jako dane empiryczne. Weryfikacja tego modelu ma zatem na celu sprawdzenie, jak szybko model jest w stanie nauczyć się liczby *ukrytych defektów* oraz mnożników czynników procesu. Dodatkowo proces weryfikacji ma pokazać, czy model jest w stanie prognozować w racjonalny sposób w różnych scenariuszach dotyczących wartości planowanych czynników procesu.

Wyniki symulacji potwierdzają, że już po pięciu iteracjach wykorzystanych do uczenia modelu model prognozuje sumę liczby defektów znalezionych z błędem względnym na poziomie 0,16 wartości rzeczywistej, a sumę liczby defektów naprawionych z błędem względnym na poziomie 0,30 wartości rzeczywistej. Te wyniki pokazują, że model jest w stanie nauczyć się swoich parametrów przy użyciu

[19] Ł. Radliński, *Przegląd publicznie dostępnych baz danych przedsięwzięć informatycznych*, Zeszyty Naukowe Uniwersytetu Szczecińskiego, „Studia Informatica" 2009, nr 22, s. 107–120.

niewielkiej liczby iteracji, a następnie generować prognozy na akceptowalnym poziomie dokładności.

Podsumowanie

Głównymi efektami badań autora są cztery sieci Bayesa do różnych zastosowań w dziedzinie inżynierii oprogramowania wraz z wynikami towarzyszących im analiz:

1. Integracja modeli do analizy kompromisu między kluczowymi zmiennymi przedsięwzięcia informatycznego w postaci *modelu produktywności*.
2. Szacowanie wskaźników produktywności i defektów na podstawie czynników środowiskowych.
3. Prognozowanie typów defektów skategoryzowanych pod względem ich uciążliwości.
4. Uczący się model iteracyjnego testowania i naprawiania defektów.

Zbudowane modele wraz z towarzyszącymi im analizami potwierdzają zasadność hipotez postawionych we wcześniejszej części. Do pozostałego wkładu autorskiego w odniesieniu do istniejących i nowych modeli zaliczyć można: dyskusję na temat możliwości parametryzacji modeli, dyskusję na temat zastosowania podejścia przyczynowego lub wskaźnikowego w modelowaniu zagregowanych wartości, dyskusję na temat zastosowania dynamicznej dyskretyzacji w zmiennych numerycznych w modelach, dyskusję na temat możliwości uwzględnienia nowych danych w modelach, przegląd najnowszych danych empirycznych wykorzystanych przy budowie modeli.

Modele opracowane w ramach rozprawy mogą być zastosowane w praktyce. Do ich budowy zostały wykorzystane zarówno dostępne dane empiryczne, jak i wiedza ekspercka. Struktury tych modeli pozwalają na łatwe ich dostosowanie do indywidualnych potrzeb przez dodawanie i usuwanie zmiennych lub zmianę wpływu danej zmiennej na inne zmienne. Kalibracja modelu może zostać dokonana „ręcznie" (wyniki ankiety) – w celu odzwierciedlenia wiedzy eksperckiej lub niewielkiego zbioru danych, albo „automatycznie" (na przykład popularnym algorytmem maksymalizacji wartości oczekiwanej) – w celu odzwierciedlenia zależności z większego zbioru danych. Możliwość przeprowadzania rozbudowanych symulacji „co-jeśli" i porównywania różnych scenariuszy, analiz wrażliwości i poszukiwania celu stanowić może istotną pomoc przy podejmowaniu decyzji przez kierowników projektów.

Literatura

Agena, *AgenaRisk. Bayesian Network Software Tool*, 2008, www.agenarisk.com, 1.06.2008.

Akiyama F., *An Example of Software System Debugging*, w: *Proc. Int. Federation for Information Processing Congress*, vol. 71, Ljubljana 1971.

Boehm B., *Software Engineering Economics*, Prentice Hall 1981.

Fenton N., Marsh W., Neil M., Cates P., Forey S., Tailor M., *Making Resource Decisions for Software Projects*, w: *Proc. 26th Int. Conf. on Software Engineering*, IEEE Computer Society, Washington, D.C. 2004.

Fenton N., Neil M., Marsh W., Hearty P., Radliński Ł., Krause P., *On the effectiveness of early life cycle defect prediction with Bayesian Nets*, "Empirical Software Engineering" 2008, vol. 13, no. 5.

Fenton N., Neil M., *Measuring your risks*, Agena White Paper, 2005, www.agenarisk.com, 1.06.2008.

Fenton N.E., Neil M., *A Critique of Software Defect Prediction Models*, "IEEE Transactions on Software Engineering" 1999, vol. 25, no. 5.

Fenton N.E., Neil M., Marsh W., Krause P., Mishra R., *Predicting Software Defects in Varying Development Lifecycles using Bayesian Nets*, "Information and Software Technology" 2007, vol. 43, no. 1.

Fenton N.E., Neil M., *Software Metrics: Roadmap*, w: *The Future of Software Engineering*. *Proc. 22nd Int. Conf. on Software Engineering*, ed. A. Finkelstein, ACM Press, 2000.

Goel A.L., Okumoto K., *An Analysis of Recurrent Software Failures in Real-Time Control System*, w: *Proc. ACM Annual Technology Conference*, Washington, D.C. 1978.

Halstead M.H., *Elements of Software Science*, Elsevier North-Holland, New York 1977.

ISBSG, *Estimating, Benchmarking & Research Suite Release 9*, International Software Benchmarking Standards Group, 2005, www.isbsg.org, 1.08.2007.

Jelinski Z., Moranda P., *Software Reliability Research*, w: *Statistical Computer Performance Evaluation*, ed. W. Freiberger, Academic Press, New York 1972.

Jones C., *Software Quality in 2002: A Survey of the State of the Art*, Software Productivity Research, Inc., 2002.

Merriam-Webster's Online Dictionary, 2008, www.merriam-webster.com, 1.06.2008.

Neil M., Tailor M., Marquez D., *Inference in hybrid Bayesian networks using dynamic discretization*, "Statistics and Computing" 2007, vol. 17, no. 3.

Putnam L.H., *A general empirical solution to the macro software sizing and estimating problem*, "IEEE Transactions on Software Engineering" 1978, vol. 4, no. 4.

Radliński Ł., Fenton N., Marquez D., *Estimating Productivity and Defects Rates Based on Environmental Factors*, w: *Information Systems Architecture and Technology: Models of the Organisation's Risk Management*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2008.

Radliński Ł., Fenton N., Neil M., Marquez D., *Improved Decision-Making for Software Managers Using Bayesian Networks*, w: *Proc. 11th IASTED Int. Conf. Software Engineering and Applications*, Cambridge, MA 2007.

Radlinski L., *Improved Software Project Risk Assessment Using Bayesian Nets*, Ph.D. Thesis, Queen Mary, University of London 2008.
Radliński Ł., *Przegląd publicznie dostępnych baz danych przedsięwzięć informatycznych*, Zeszyty Naukowe Uniwersytetu Szczecińskiego, „Studia Informatica" 2009, nr 22.
Rainer A., Hall T., *Identifying the causes of poor progress in software projects*, w: *Proc. 10th Int. Symposium on Software Metrics*, Sep. 2004.
Sassenburg J.A., *Design of a Methodology to Support Software Release Decisions (Do the numbers Really Matter?)*, Ph.D. Thesis, University of Groningen 2006.
Webster's Online Dictionary, 2008, www.websters-online-dictionary.org, 1.06.2008.

IMPROVED SOFTWARE PROJECT RISK ASSESSMENT USING BAYESIAN NETS

Summary

Empirical software engineering models typically focus on predicting development effort or software quality but not both. Using Bayesian Nets (BNs) as causal models, researchers have recently attempted to build models that incorporate relationships between functionality, effort, software quality, and various process variables. The thesis analyses such models and, as part of a new validation study, identifies their strengths and weaknesses. A major weakness is their inability to incorporate prior local productivity and quality data, which limits their applicability in real software projects. The main hypothesis is that it is possible to build BN models that overcome these limitations without compromising their basic philosophy. In particular, the thesis shows we can build BNs that capture known trade-offs and can be tailored to individual company needs.

The new model, called the *productivity model*, is developed by using the results of the new validation of the existing models, together with various other analyses. These include: the results of applying various statistical methods to identify relationships between a range of variables using publicly available data on software projects; analyses of other studies; expert knowledge. The new model is also calibrated using the results of an extensive questionnaire survey of experts in the area.

The thesis also makes a number of other novel contributions to improved risk assessment using BNs, including a model which predicts the proportions of different types of defects likely to be left in software after testing and a learning model for predicting the number of defects found and fixed in successive testing iterations.

Translated by Łukasz Radliński