ZESZYTY NAUKOWE UNIWERSYTETU SZCZECIŃSKIEGO

NR 611 STUDIA INFORMATICA NR 26 2010

ŁUKASZ RADLIŃSKI

TECHNIKI PROGNOZOWANIA NAKŁADÓW PROJEKTOWYCH I JAKOŚCI OPROGRAMOWANIA W PROJEKTACH IT

Wprowadzenie

Prognozowanie nakładów projektowych i jakości[1] oprogramowania jest wymagającym zadaniem w inżynierii oprogramowania zarówno dla naukowców, jak i praktyków. Wczesne modele parametryczne[2] zostały zbudowane prawie czterdzieści lat temu, jednak nadal są używane w niektórych firmach IT oraz w celach naukowych. Obecnie zaobserwować możemy wzrost zainteresowania bardziej „inteligentnymi" technikami z zakresu sztucznej inteligencji lub zbliżonych. W większości analiz autorzy informują, że takie nowoczesne techniki są lepsze od modeli parametrycznych pod względem dokładności prognoz. Z praktycznego jednak punktu widzenia ważne jest podkreślenie, że żadna z tych technik nie jest idealna. Autorzy często wymieniają zalety technik przez nich

[1] W praktyce nie istnieje ogólnie uznana miara jakości oprogramowania. Uznaje się, że na jakość składa się wiele cech, wśród których najczęściej ujmowane są w modelach prognostycznych następujące: defekty (liczba defektów lub gęstość defektów), niezawodność (czas między awariami, liczba awarii w danym przedziale czasu), identyfikacja komponentów oprogramowania (np. modułów, klas, metod) podatnych na błędy, poziom zadowolenia użytkownika.

[2] F. Akiyama, *An Example of Software System Debugging*, Proc. Int. Federation for Information Processing Congress, vol. 71, Ljubljana, 1971, s. 353–379; Z. Jelinski, P. Moranda, *Software Reliability Research*, w: *Statistical Computer Performance Evaluation*, ed. W. Freiberger, Academic Press, New York 1972, s. 465–484; 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. 345–361.

wykorzystywanych, pomijając ich wady w odniesieniu do możliwego zastosowania w przemyśle.

Przemysłowe wykorzystanie technik prognozowania nakładów i jakości różni się od użycia naukowego. Wiele z nich wymaga dostępu do dużych, kompletnych i spójnych zbiorów danych. Firmy IT rzadko mają takie dane, zwykle dysponują wiedzą ekspercką. Wiedza taka jest mocno subiektywna, trudno więc na jej podstawie zbudować uniwersalne modele prognostyczne wiedzy. Łatwiejsze jest zbudowanie takich modeli dla konkretnej firmy działającej w określonym środowisku. Z punktu widzenia przemysłu jest to lepsze rozwiązanie, gdyż taki model ujmuje specyficzne potrzeby firmy oraz cechy jej otoczenia. Istnieje zatem zapotrzebowanie na taką technikę, która umożliwi połączenie wiedzy eksperckiej i empirycznych danych. Sieci Bayesa wydają się zaspokajać te warunki, mają także inne zalety, które występują tylko częściowo w innych technikach.

W punkcie 1 niniejszego artykułu została przeprowadzona analiza zastosowania różnych technik prognozowania nakładów projektowych i jakości oprogramowania. W drugim zostały porównane różne techniki oraz uzasadniono wybór sieci Bayesa do budowy modeli prognostycznych możliwych do wykorzystania w firmach IT. W trzecim zaś zawarto podstawowe informacje o sieciach Bayesa oraz pokrótce omówiono ich zastosowania w dziedzinie inżynierii oprogramowania.

1. Przegląd zastosowań technik prognozowania nakładów i jakości

W niniejszym artykule poddane analizie zostały zastosowania następujących technik:

- modele parametryczne (*parametric models* – PM) najczęściej zbudowane w wyniku regresji wielorakiej,
- dynamika systemowa (*system dynamics* – SD),
- sztuczne sieci neuronowe (*artificial neural networks* – NN),
- zbiory rozmyte (*fuzzy sets* – FS),
- zbiory przybliżone (*rough sets* – RS),
- szacowanie przez analogię (*estimation by analogy* – EA lub *case-based reasoning* – CBR),
- drzewa decyzyjne (*decision trees* – DT lub *classification and regression trees* – CART),

– indukcyjne programowanie logiki (*inductive logic programming* – ILP),
– algorytmy ewolucyjne (*evolutionary algorithms* – EVA), wśród których najbardziej popularne są algorytmy genetyczne (*genetic algorithms* – GA),
– wektory nośne (*support vector machines* – SVM),
– analiza Bayesowska (*Bayesian analysis* – BA),
– sieci Bayesa (*Bayesian nets* – BN).

W tabeli 1 przedstawiono zestawienie najważniejszych publikacji wykorzystanych w analizie zastosowań technik prognozowania nakładów i jakości oprogramowania. Cała analiza objęła 65 publikacji, w artykule jednak zostaną omówione te, w których wykorzystano co najmniej dwie techniki.

W większości przypadków autorzy skupiają się na ocenie danej techniki na podstawie dokładności prognozy osiąganej przez model zbudowany przy jej użyciu. Doświadczenia wskazują[3], że ta dokładność zależy najbardziej od zestawu użytych danych. Dlatego w niniejszej pracy przeanalizowane zostały inne cechy technik, które powinny być wzięte pod uwagę przy wyborze najlepszej z nich.

[3] V.U.B. Challagulla, F.B. Bastani, Y. I-Ling, R.A. Paul, *Empirical assessment of machine learning based software defect prediction techniques*, Proc. 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems 2005, s. 263–270; K. Srinivasan, D. Fisher, *Machine learning approaches to estimating software development effort*, "IEEE Transactions on Software Engineering" 1995, vol. 21, no. 2, s. 126–137; B. Stewart, *Predicting project delivery rates using the Naive–Bayes classifier*, "Journal on Software Maintenance and Evolution: Research and Practice" 2002, vol. 14, s. 161–179.

Tabela 1

Zestawienie metod wykorzystywanych do prognozowania jakości oprogramowania

Autor	Prognozowana wielkość	Techniki
Hewett i Kijsanayothin[1)]	czas naprawy defektów	BA, DT, EA, NN, SVM
Zhang i Tsai[2)]	różne aspekty nakładów i jakości	BA, EA, EVA, NN, ILP, inne
Challagulla i in.[3)]	defekty, podatność na błędy	BA, EA, PM, NN
Chulani i in.[4)]	nakłady	BA, PM
van Koten[5)]	nakłady	BA, PM
Stewart[6)]	nakłady	BN, DT, NN
Fenton i in.[7)]	defekty	BN, PM
Bai i in.[8)]	niezawodność	BN, PM
Huang i Chiu[9)]	nakłady	DT, EA+EVA, NN, PM
Ceylan i in.[10)]	podatność na błędy	DT, NN
Srinivasan i Fisher[11)]	nakłady	DT, NN
Gokhale i Lyu[12)]	defekty	DT, PM
Schröter i in.[13)]	podatność na błędy	DT, PM, SVM
Shepperd i Kadoda[14)]	nakłady	EA, DT, PM, NN
Chiu i Huang[15)]	nakłady	EA, EVA
Mair i in.[16)]	nakłady	EA, ILP, NN, PM
Shepperd i Schofield[17)]	nakłady	EA, PM
Walkerden i Jeffery[18)]	nakłady	EA, PM
Li i Ruhe[19)]	nakłady	EA, RS
Gray i MacDonell[20)]	nakłady	FS, NN, PM
Stefanowski[21)]	nakłady	ILP, RS
de Barcenos Tronto i in.[22)]	nakłady	NN, PM
Dohi i in.[23)]	niezawodność	NN, PM
Khoshgoftaar[24)]	defekty	NN, PM
Sitte[25)]	niezawodność	NN, PM
Ramanna[26)]	nakłady	NN, RS
Hochman i in.[27)]	podatność na błędy	NN+EVA

Objaśnienia: [1)] R. Hewett, P. Kijsanayothin, *On modeling software defect repair time*, "Empirical Software Engineering" 2009, vol. 12, no. 2, s. 165–186. [2)] D. Zhang, J.J.P. Tsai, *Machine Learning and Software Engineering*, "Software Quality Journal" 2003, vol. 11, no. 2, s. 87–119. [3)] V.U.B. Challagulla, F.B. Bastani, Y. I-Ling, R.A. Paul, *Empirical assessment*... [4)] S. Chulani, B. Boehm, B. Steece, *Bayesian Analysis of Empirical Software Engineering Cost Models*, "IEEE Transactions on Software Engineering" 1999, vol. 25, no. 4 s. 573–583. [5)] C. van Koten, *Bayesian statistical models for predicting software development effort*, Discussion Paper 2005/08, Department of Information Science, University of Otago, Dunedin, New Zealand 2005. [6)] B. Stewart, *Predicting project*... [7)] 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. [8)] C.G. Bai, Q.P. Hu, M. Xie, S.H. Ng, *Software failure prediction based on a Markov Bayesian network model*, "Journal of Systems and Software" 2005, vol. 74, no. 3, s. 275–282. [9)] S.-J. Huang, N.-H. Chiu, *Optimization of analogy weights by genetic algorithm for software effort estimation*, "Information and Software Technology" 2006, vol. 48, no. 11, s. 1034–1045. [10)] E. Ceylan, F.O. Kutlubay, A.B. Bener, *Software Defect Identification Using Machine Learning Techniques*, Proc. 32nd EUROMICRO Conf. on Software Engineering and Advanced Applications, Crotia 2006, s. 240–247. [11)] K. Srinivasan, D. Fisher, *Machine learning approaches...* [12)] S.S. Gokhale, M.R. Lyu, *Regression Tree Modelling for the Prediction of Software Quality*, Proc. ISSAT Int. Conf. on Reliability and Quality in Design, Anaheim 1997, s. 31–36. [13)] A. Schröter, T. Zimmermann, A. Zeller, *Predicting component failures at design time*, Proc. 2006 ACM/IEEE Int. Symposium on Empirical Software Engineering, ACM, New York 2006, s. 18–27. [14)] M. Shepperd, G. Kadoda, *Comparing Software Prediction Techniques Using Simulation*, "IEEE Transactions on Software Engineering" 2001, vol. 27, no. 11, s. 1014–1022. [15)] N. Chiu, S. Huang, *The adjusted analogy-based software effort estimation based on similarity distances*, "Journal of Systems and Software" 2007, vol. 80, no. 4, s. 628–640. [16)] C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd, S. Webster, *An investigation of machine learning based prediction systems*, "Journal of Systems and Software" 2000, vol. 53, no. 1, s. 23–29. [17)] M. Shepperd, C. Schofield, *Estimating Software Project Effort Using Analogies*, "IEEE Transactions on Software Engineering" 1997, vol. 23, no. 11, s. 736–743. [18)] F. Walkerden, R. Jeffery, *An Empirical Study of Analogy-based Software Effort Estimation*, "Empirical Software Engineering" 1999, vol. 4, no. 2, s. 135–158. [19)] J. Li, G. Ruhe, *Analysis of attribute weighting heuristics for analogy-based software effort estimation method AQUA+*, "Empirical Software Engineering" 2008, vol. 13, no. 1, s. 63–96. [20)] A. Gray, S. MacDonell, *Applications of fuzzy logic to software metric models for development effort estimation*, Proc. 1997 Annual Meeting North American Fuzzy Information Processing Society 1997, s. 394–399. [21)] J. Stefanowski, *An empirical study of using rule induction and rough sets to software cost estimation*, "Fundamenta Informaticae" 2006, vol. 71, no. 1, s. 63–82. [22)] I.F. de Barcelos Tronto, J.D. da Silva, N. Sant'Anna, *An investigation of artificial neural networks based prediction systems in software project management*, "Journal of Systems Software" 2008, vol. 81, no. 3, s. 356–367. [23)] T. Dohi, Y. Nishio, S. Osaki, *Optimal software release scheduling based on artificial neural networks*, "Annals of Software Engineering" 1999, vol. 8, no. 1–4, s. 167–185. [24)] T. Khoshgoftaar, A. Pandya, D. Lanning, *Application of neural networks for predicting program faults*, "Annals of Software Engineering" 1995, vol. 1, no. 1, s. 141–154. [25)] R. Sitte, *Comparison of software-reliability-growth predictions: neural networks vs parametric-recalibration*, "IEEE Transactions on Reliability" 1999, vol. 48, no. 3, s. 285–291. [26)] S. Ramanna, *Rough Neural Network for Software Change Prediction*, Proc. 3rd Int. Conf. on Rough Sets and Current Trends in Computing, LNCS, vol. 2475, Springer-Verlag, London 2002, s. 602–609. [27)] R. Hochman, E. Allen, J. Hudepohl, T. Khoshgoftaar, *Evolutionary neural networks: A robust approach to software reliability problems*, Proc. 8th Int. Symp. on Software Reliability Engineering, IEEE Computer Society, Washington, DC 1997, s. 13.

Zhang i Tsai[4] ocenili wybrane techniki, wykorzystując różne kryteria, jak: wymagana wiedza dziedzinowa (WD), dane uczące (DU), zalety (Z) i wady (W)

– BA – WD: wymagana w zakresie prawdopodobieństw *a priori*. DU: stopniowo zmniejszają lub podwyższają szacowane prawdopodobieństwo. Z: elastyczne w uczeniu funkcji docelowej. Prognozy probabilistyczne. W: wymagane liczne prawdopodobieństwa *a priori*. Koszt obliczeń.
– DT – WD: niewymagana. DU: odpowiednie dane wymagane w celu uniknięcia przeuczenia. Tolerowane braki danych. Z: efektywne przy zaszumionych danych. Umożliwiają nauczenie wyrażeń rozłącznych. W: przeuczanie.
– EA – WD: niewymagana. DU: dużo potrzebnych danych. Z: szybkie uczenie. Możliwe uczenie złożonych funkcji. Brak strat informacji. W: wolna w czasie wyszukiwania. Przekleństwo wymiarowości.
– EVA – WD: niewymagana. DU: niewymagana (część danych testowych może być potrzebna do sprawdzenia dopasowania). Z: przeznaczone do sytuacji, w których szacowane funkcje są złożone. Algorytmy mogą być łatwo zrównoleglone. W: zatłoczenie. Zasobożerność.
– NN – WD: niewymagana. DU: dużo potrzebnych danych. Z: dobrze sobie radzi z błędami w danych treningowych. Możliwe nauczenie złożonych funkcji (nieliniowych, ciągłych). Równoległy, rozproszony proces uczenia. W: wolne proces uczenia i osiąganie zbieżności. Wielokrotne lokalne minima na powierzchni błędów. Przeuczanie.
– ILP – WD: wymagana. DU: podzielone na pozytywne i negatywne przypadki. Z: czytelna reprezentacja nauczonej funkcji. Indukcja sformułowana jako odwrotność dedukcji. Wyszukiwanie wspierane wiedzą dziedzinową. W: mało efektywne przy zaszumionych danych. Trudne przeszukiwanie w przypadkach ogólnych. Wzrost wiedzy dziedzinowej prowadzi do wzrostu złożoności przestrzeni hipotez. Brak gwarancji na znalezienie najmniejszego lub najlepszego zestawu reguł.

Stewart[5] odkrył, że NN funkcjonowały bardzo dobrze z prawie każdym analizowanym zestawem danych, były jednak podatne na przeuczanie. Ponadto wykazał, że naiwny klasyfikator Bayesa (rodzaj prostej sieci) nie jest wrażliwy na szum w danych, jest mniej podatny na przeuczenie niż NN oraz, że może być wykorzystywany jako alternatywa dla DT i NN.

[4] D. Zhang, J.J.P. Tsai, *Machine Learning*...

[5] B. Stewart, *Predicting project*...

Schepperd i Kadoda[6] zaproponowali zestaw reguł do wzięcia pod uwagę przy decydowaniu o użyciu konkretnej techniki, na przykład:

– Regresja krokowa wsteczna jest preferowana dla zmiennych zależnych ciągłych zgodnych z rozkładem normalnym, kiedy zbiór danych nie zawiera elementów odstających (*outliers*) i nie charakteryzuje się współliniowością.
– EA jest zalecana do dyskretnej zmiennej zależnej i gdy zbiór danych zawiera elementy odstające i charakteryzuje się współliniowością.
– Techniki uczenia maszynowego, które zawsze lepiej funkcjonują przy większych zbiorach danych.

Mair i inni[7] przeprowadzili rozbudowaną analizę porównawczą różnych technik, w wyniku której uszeregowali techniki na podstawie trzech kryteriów (w nawiasach wskazania od najlepszej do najgorszej):

– dokładność: (1) NN, (2 *ex aequo*) PM i EA, (4) ILP,
– stopień wyjaśnienia: (1 *ex aequo*) EA, PM i ILP, (4) NN,
– konfigurowalność: (1 *ex aequo*) EA i PM, (3) ILP, (4) NN.

2. Porównanie metod prognozowania

Po przeglądzie zastosowań analizowanych technik następne etapy analizy objęły ocenę przydatności poszczególnych technik (tabela 2) i wybór techniki najlepiej dopasowanej do prognoz w przemyśle IT. Ocena została dokonana na podstawie cech technik oraz wyników ich zastosowań omówionych w poprzedniej części.

[6] M. Shepperd, G. Kadoda, *Comparing Software Prediction*...

[7] C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd, S. Webster, *An investigation of machine learning based*...

Tabela 2

Zestawienie istotnych cech użyteczności analizowanych technik wykorzystywanych w inżynierii oprogramowania

Cecha / Technika	Możliwość ujęcia zależności przyczynowo-skutkowych	Możliwość ujęcia niepewności	Intuicyjność modelu	Łatwość adaptacji	Możliwość budowy modelu jedynie na podstawie wiedzy eksperta
SD	BW	BN	BW	Ś	BW
NN	Nie	Ś	N	Ś	BN
FS	Nie	W	W	Ś	Ś
RS	W	BW	BW	W	Ś
EA	Nie	Ś	Ś	Ś	BN
DT	W	BW	W	Ś	BW
ILP	BW	Ś	BW	W	BW
EVA	Nie	W	Ś	Ś	BW
SVM	Nie	W	Ś	Ś	BN
BA	Nie	BW	W	Ś	Ś
BN	BW	BW	BW	W	BW

Legenda: „BW" – bardzo wysoka, „W" – wysoka, „Ś" – średnia, „N" – niska, „BN" – bardzo niska.

Źródło: opracowanie własne na podstawie przytoczonej literatury.

Przeprowadzona analiza porównawcza zastosowań poszczególnych technik doprowadziła do następujących spostrzeżeń:

1. Żadna z analizowanych technik nie daje najlepszych wyników w każdym przypadku. Autorzy, którzy używali jednej techniki, zwykle wykazywali sukces w swoich badaniach. Autorzy, którzy porównywali wyniki osiągnięte różnymi technikami w tej samej analizie, zwykle byli w stanie wskazać technikę dającą najlepsze wyniki. Jednak w wielu przypadkach autorzy korzystający z tych samych technik w innej analizie osiągali najlepsze wyniki inną techniką. W konsekwencji niektórzy badacze[8] zaproponowali reguły przydatne w wyborze techniki do rozwiązania konkretnego problemu.

[8] M. Shepperd, G. Kadoda, *Comparing Software Prediction...*

2. Prawie wszyscy autorzy[9], którzy korzystali z modeli parametrycznych i innych technik, osiągali lepsze wyniki tymi innymi technikami. Jedynym wyjątkiem jest praca Shepperda i Kadody[10], w której autorzy zastrzegają, że tylko w szczególnych przypadkach modele parametryczne dają lepsze wyniki od innych technik.
3. Większość analizowanych technik jest zautomatyzowanych – wynikiem jest model prognostyczny całkowicie lub w większości zależny od dostępnych danych. Techniki te zwykle wymagają pełnych, dużych i spójnych zbiorów danych, które w warunkach przemysłowych mogą nie być dostępne. W takich przypadkach bardzo cenna jest możliwość budowy modelu opartego na wielu źródłach danych, informacji czy wiedzy: wynikach analiz empirycznych danych, wiedzy eksperckiej opartej na doświadczeniu lub innych częściowo raportowanych danych. Jednak część technik (NN, EA, SVM) nie umożliwia wykorzystania tak różnych źródeł.
4. Większość analizowanych technik nie pozwala na jawne ujęcie niepewności w odniesieniu do modelowanej części rzeczywistości. Odzwierciedlenie złożonego systemu jest nieuchronnie związane z jego uproszczeniem w postaci modelu. Niektóre elementy rzeczywistości, takie jak zmienne czy powiązania między nimi, nie są włączane do modelu. Ponadto przebieg projektów IT jest bardzo zależny od ludzi, których zachowanie rzadko, jeśli w ogóle, możliwe jest do ujęcia z całkowitą pewnością.
5. Większość analizowanych technik wytwarza modele, które albo nie są intuicyjne dla ekspertów IT (NN), albo są czytelne jedynie w prostych przypadkach z kilkoma zmiennymi i kilkoma możliwymi ich wartościami (DT, ILP, RS).

[9] C.G. Bai, Q.P. Hu, M. Xie, S.H. Ng, *Software failure prediction based...*; S. Chulani, B. Boehm, B. Steece, *Bayesian Analysis...*; I.F. de Barcelos Tronto, J.D. da Silva, N. Sant'Anna, *An investigation of artificial neural networks based...*; T. Dohi, Y. Nishio, S. Osaki, *Optimal software...*; S.S. Gokhale, M.R. Lyu, *Regression Tree Modelling...*; T. Khoshgoftaar, A. Pandya, D. Lanning, *Application of neural networks...*; M. Shepperd, C. Schofield, *Estimating Software Project Effort Using Analogies...*; C. van Koten, *Bayesian statistical models...*

[10] M. Shepperd, G. Kadoda, *Comparing Software Prediction...*

6. Porównania wyników zastosowań niektórych technik nie są publikowane. Na przykład autorzy wykorzystujący SD[11] nie publikują porównań osiągniętych wyników z wynikami osiągniętymi innymi technikami. Zapewne jest to spowodowane czasochłonnym opracowywaniem modelu za pomocą SD, a w konsekwencji brakiem nakładów na opracowanie modeli za pomocą innych technik. Ponadto, bezpośrednie porównania BN, FS i RS z innymi technikami są bardzo rzadkie.

Następny krok analizy to wybór „najlepszej" techniki proponowanej do użycia przy budowie modeli predykcyjnych w dziedzinie zarządzania projektami IT, szczególnie do prognozowania nakładów i jakości. Przeprowadzona analiza wykazała, że w zasadzie wszystkie techniki mogą być wykorzystywane do budowy takich modeli – autorzy twierdzą, że osiągnęli dobre wyniki tymi technikami. Zmiana punktu widzenia ze zorientowanego naukowo na zorientowany przemysłowo wykazuje jednak ograniczenia większości tych technik. Główny problem stanowi dostępność danych. W praktyce zbiory danych o minionych projektach w firmach IT są znacząco niekompletne, co ogranicza lub uniemożliwia użycie technik zautomatyzowanych. Zatem jednym z kluczowych wymagań wobec „najlepszej" techniki jest jak najmniejsza zależność od danych empirycznych. Preferowana jest możliwość integracji wiedzy eksperckiej z wynikami analiz opartych na danych empirycznych.

Na podstawie przeprowadzonego porównania sieci Bayesa wydają się najlepiej dostosowane do takich potrzeb. Dodatkowo mają kolejne zalety, przy czym niektóre z nich odnoszą się również do innych technik. Są to:

– możliwość przeprowadzania analizy kompromisu (*trade-off*) między kluczowymi czynnikami projektu: nakładami, rozmiarem i jakością, co jest możliwa tylko wtedy, gdy technika udostępnia wnioskowanie wprzód i wstecz,
– brak stałych osobnych list predyktorów i zmiennych zależnych oraz możliwość dokonywania prognoz przy niepełnych danych – kiedy użytkownik wprowadza obserwację do modelu (przypisuje wartość do zmiennej), zmien-

[11] J.S. Collofello, Z. Yang, J.D. Tvedt, D. Merrill, J. Rus, *Modelling Software Testing Processes*, Proc. IEEE 15th Int. Phoenix Conference on Computers and Communications, 1995; D. Pfahl, A. Al-Emran, G. Ruhe, *A System Dynamics Simulation Model for Analyzing the Stability of Software Release Plans: Research Sections*, Software Process: Improvement and Practice 2007, vol. 12, no. 5, s. 475–490; D.M. Raffo, W. Harrison, J. Vandeville, *Coordinating Models and Metrics to Manage Software Projects*, "Software Process: Improvement and Practice" 2000, vol. 5, no. 2–3, s. 159–168.

na ta staje się predyktorem przy szacowaniu zmiennej bez wprowadzonej obserwacji (zależnej),
– jawne ujęcie niepewności przez definicję wszystkich zmiennych rozkładami prawdopodobieństw,
– łatwe łączenie zmiennych jakościowych i ilościowych.

3. Sieci Bayesa i ich zastosowania

Sieć Bayesa (BN) to skierowany acykliczny graf, który zawiera zbiór zmiennych losowych i skierowanych połączeń między parami zmiennych. BN może być postrzegana z perspektywy funkcjonalnej jako graf ilustrujący powiązania między zmiennymi, co powoduje, że taki model jest czytelny dla ekspertów IT, którzy rzadko są specjalistami z dziedziny BN, sztucznej inteligencji czy statystyki. Jednocześnie BN wraz ze swoją formalną matematyczną definicją zmiennych pozwala na ilościowe odzwierciedlenie modelowanego fragmentu rzeczywistości i rygorystyczne wnioskowanie oparte na rachunku prawdopodobieństwa, szczególnie na twierdzeniu Bayesa[12].

Zależnie od celu analizy i dostępności danych empirycznych trzy główne typy sieci ze względu na topologię są wykorzystywane w inżynierii oprogramowania:

1. Naiwny klasyfikator Bayesowski (*naïve Bayesian classifier* – NBC), który charakteryzuje się strukturą gwiazdy i nie zawiera związków przyczynowo-skutkowych między zmiennymi; zwykle budowany jest automatycznie na podstawie dostępnych danych.
2. Przyczynowa sieć Bayesa (*causal Bayesian net* – CBN), która zawiera związki przyczynowo-skutkowe między zmiennymi; może być zbudowana automatycznie na podstawie dostępnych danych, ręcznie, uwzględniając wiedzę ekspercką, lub integrować różne źródła.
3. Dynamiczna sieć Bayesa (*dynamic Bayesian net* – DBN), która jest złożona z sekwencyjnie połączonych CBN, w celu odzwierciedlenia dynamiki modelowanego procesu.

[12] T. Bayes, *An essay towards solving a Problem in the Doctrine of Chances. By the late Rev. Mr. Bayes, F.R.S. communicated by Mr. Price, in a letter to John Canton, A.M.F.R.S.*, "Philosophical Transactions of the Royal Society of London" 1763, vol. 53, s. 370–418.

W tabeli 3 zestawiono najbardziej znane zastosowania BN w inżynierii oprogramowania. Szczegółowa analiza porównawcza wybranych BN została dokonana we wcześniejszej pracy[13].

Tabela 3

Zestawienie zastosowań sieci Bayesa w dziedzinie inżynierii oprogramowania

Autor	Główny problem analizowany	Typ sieci	Zweryfikowany na danych empirycznych
Stewart[1)]	nakłady, produktywność	NBC	tak
Pai i in.[2)]	niezależna walidacja i weryfikacja	CBN	nie
Fenton i in.[3)]	kompromis między nakładami, wielkością i jakością	CBN	nie
Fenton i in.[4)]	defekty, częściowo nakłady	CBN/DBN	tak
Radliński i in.[5)]	kompromis między nakładami, wielkością i jakością	CBN	nie
Hearty i in.[6)]	szybkość projektu (produktywność)	DBN	tak
Fenton i in.[7)]	defekty	DBN	nie
Bibi i Stamelos[8)]	nakłady	DBN	nie
Bai i in.[9)]	awarie	DBN	tak
Cockram[10)]	efektywność inspekcji	CBN	tak
Wooff i in.[11)]	proces testowania	CBN	tak

Objaśnienia: [1)] B. Stewart, *Predicting project...* [2)] G.I. Pai, J.B. Dugan, K. Lateef, *Bayesian Networks applied to Software IV&V*, Proc. 29th Annual IEEE/NASA Software Engineering Workshop 2005, s. 293–304. [3)] N. Fenton, W. Marsh, M. Neil, P. Cates, S. Forey, M. Tailor, *Making Resource Decisions for Software Projects*, Proc. 26th Int. Conf. on Software Engineering, IEEE Computer Society, Washington, DC 2004, s. 397–406. [4)] Tamże oraz 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. [5)] Ł. Radliński, N. Fenton, M. Neil, D. Marquez, *Improved Decision-Making for Software Managers Using Bayesian Networks*, Proc. 11th IASTED International Conference Software Engineering and Applications, Cambridge, MA 2007, s. 13–19; L. Radlinski, *Improved Software Project Risk Assessment Using Bayesian Nets*, Ph.D. Thesis, Queen Mary, University of London, London 2008. [6)] P. Hearty, N. Fenton, D. Marquez, M. Neil, *Predicting Project Velocity in XP using a Learning Dynamic Bayesian Network Model*, "IEEE Transaction on Software Engineering" 2009, vol. 37, no. 1, s. 124–137. [7)] N. Fenton, P. Hearty, M. Neil, Ł. Radliński, *Software Project and Quality Modelling Using Bayesian Networks*, w: *Artificial Intelligence Applications for Improved Software*

[13] Ł. Radliński, *Przegląd sieci Bayesa do szacowania ryzyka w inżynierii oprogramowania*, Zeszyty Naukowe Uniwersytetu Szczecińskiego nr 476, Studia Informatica nr 21, Szczecin 2009, s. 119–129.

Engineering Development: New Prospects, ed. F. Meziane, S. Vadera, IGI-Global 2009. [8)] S. Bibi, I. Stamelos, *Software Process Modeling with Bayesian Belief Networks*, Proc. of 10th International Software Metrics Symposium, Chicago 2004. [9)] C.G. Bai, Q.P. Hu, M. Xie, S.H. Ng, *Software failure prediction based...* [10)] T. Cockram, *Gaining Confidence in Software Inspection Using a Bayesian Belief Model*, "Software Quality Journal" 2001, vol. 9, no. 1, s. 31–42. [11)] D.A. Wooff, M. Goldstein, F.P.A. Coolen, *Bayesian Graphical Models for Software Testing*, "IEEE Transactions on Software Engineering" 2002, vol. 28, no. 5, s. 510–525.

Źródło: opracowanie własne na podstawie przytoczonej literatury.

Z teorii zarządzania oraz z praktyki zarządzania przedsięwzięciami IT znany jest związek mówiący o tym, że

– z perspektywy nakładów: zwiększenie nakładów na dany projekt prowadzi do polepszenia jakości wytwarzanego produktu,
– z perspektywy jakości: w celu osiągnięcia wyższej jakości produktu potrzebne jest zwiększenie nakładów na dany projekt.

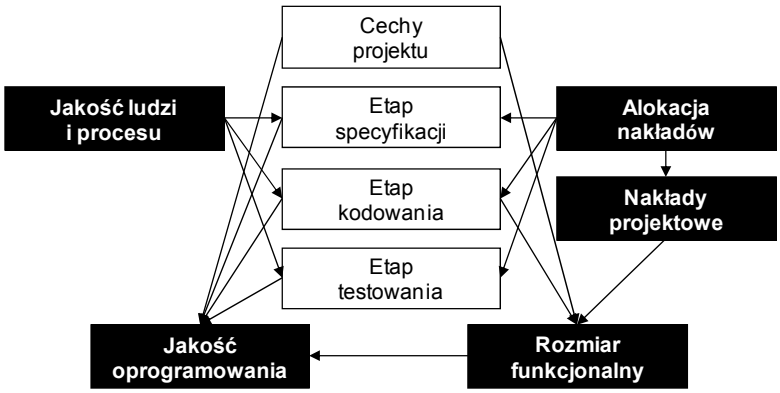
Jednak tylko nieliczne modele, włączając w to BN, potrafią ująć ten związek z obu stron, umożliwiając prognozowanie nakładów i jakości w ramach pojedynczego modelu. Model MODIST[14] jest pierwszym szerzej znanym modelem opracowanym na skalę przemysłową, który umożliwia taką analizę kompromisu między nakładami, rozmiarem i jakością. Później został opracowany ulepszony model o całkiem nowej strukturze nazwany modelem produktywności[15] (rysunek 1), który implementuje podstawowe założenie modelu MODIST i oferuje nowe możliwości niedostępne wcześniej:

– kluczowe zmienne wyrażone na skali numerycznej,
– możliwość ujęcia lokalnych wskaźników produktywności i defektów podanych przez użytkowników na podstawie minionych projektów,
– dowolne jednostki miar dla nakładów, rozmiaru, produktywności i wskaźnika defektów,
– łatwa kalibracja modelu dzięki użyciu dostarczonej ankiety,

[14] Tamże; N. Fenton, W. Marsh, M. Neil., P. Cates, S. Forey, M. Tailor, *Making Resource Decisions for Software Projects*, Proc. 26th Int. Conf. on Software Engineering, IEEE Computer Society, Washington, DC 2004.

[15] Ł. Radliński, *Przegląd sieci Bayesa...*; N. Fenton i in., *Making Resource Decisions...*; N. Fenton, Ł. Radliński, M. Neil, *Improved Bayesian Networks...*

– większa precyzja prognoz dzięki zastosowaniu dynamicznej dyskretyzacji zmiennych numerycznych[16].



Rys. 1. Schemat modelu produktywności
Źródło: opracowanie własne.

Głównym powodem wyboru BN jako techniki użytej do budowy i definicji modelu produktywności był brak dostępnego kompletnego zbioru danych empirycznych o wymaganej wielkości i jakości do innych technik. W modelu zostały zintegrowane następujące źródła danych:

– identyfikacja zmiennych: przegląd literatury, analiza wcześniejszych modeli, wiedza ekspercka,
– struktura modelu: wiedza ekspercka,
– bezwarunkowe rozkłady prawdopodobieństw dla zmiennych numerycznych: analiza statystyczna,
– bezwarunkowe rozkłady prawdopodobieństw dla zmiennych porządkowych: wiedza ekspercka,
– warunkowe rozkłady prawdopodobieństw (wagi czynników procesu i wpływ zmiennych porządkowych na numeryczne): badanie ankietowe.

Model produktywności zawiera związki przyczynowo-skutkowe między zmiennymi, co zwiększa jego intuicyjność dla użytkowników niebędących ekspertami w dziedzinie BN. Przeprowadzone liczne testy wykazały, że model

[16] N. Fenton, Ł. Radliński, M. Neil, *Improved Bayesian Networks for Software Project Risk Assessment Using Dynamic Discretisation*, w: *Software Engineering Techniques: Design for Quality*, ed. K. Sacha, "IFIP International Federation for Information Processing" 2006, vol. 227, s. 139–148.

poprawnie ujmuje wiedzę z zakresu zarządzania projektami IT i może być istotną pomocą dla kierowników projektów[17].

Podsumowanie

Przeprowadzona analiza porównawcza technik prognozowania nakładów projektowych i jakości oprogramowania wykazała, że większość z metod może być z sukcesem wykorzystywana w tym celu. Sukces ten zwykle może być osiągnięty, jeśli dostępny jest duży, kompletny i spójny zbiór danych o minionych projektach. Firmy IT często nie gromadzą takich danych, stąd należy zasugerować korzystanie z technik umożliwiających integrację wiedzy eksperckiej z danymi empirycznymi.

Sieci Bayesa pozwalają na takie połączenie i mają liczne zalety nad innymi technikami. Modele BN były wykorzystywane między innymi do prognozowania nakładów i jakości, gdzie wydaje się, że mają wysoki potencjał. Budowa modelu jako BN nie gwarantuje automatycznie sukcesu w dokonywaniu prognoz tylko dlatego, że jest to BN. Takie modele są mocno oparte na wiedzy eksperckiej, stąd należy zwrócić dużą uwagę na ich weryfikację. Ma ona krytyczne znaczenie i jest trudna do przeprowadzenia – niewielka liczba dostępnych danych silnie ogranicza możliwość przeprowadzenia weryfikacji danymi empirycznymi.

Przyszłe badania autora planowane są w kierunku rozszerzenia modeli ryzyka projektów IT przez dodanie czynników specyficznych dla danych typów projektów, umożliwienie optymalizacji procesu wytwórczego i odzwierciedlenie innych cech jakości, takich jak łatwość konserwacji zadowolenie użytkownika, wydajność i inne.

Literatura

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

Bai C.G., Hu Q.P., Xie M., Ng S.H., *Software failure prediction based on a Markov Bayesian network model*, "Journal of Systems and Software" 2005, vol. 74, no. 3.

Bayes T., *An essay towards solving a Problem in the Doctrine of Chances. By the late Rev. Mr. Bayes, F.R.S. communicated by Mr. Price, in a letter to John Canton, A.M.F.R.S.*, "Philosophical Transactions of the Royal Society of London" 1763, vol. 53.

[17] Ł. Radliński, *Przegląd sieci Bayesa*...; N. Fenton i in., *Making Resource Decisions*...; N. Fenton, Ł. Radliński, M. Neil, *Improved Bayesian Networks*...

Bibi S., Stamelos I., *Software Process Modeling with Bayesian Belief Networks*, Proc. of 10th International Software Metrics Symposium, Chicago 2004.

Ceylan E., Kutlubay F.O., Bener A.B., *Software Defect Identification Using Machine Learning Techniques*, Proc. 32nd EUROMICRO Conf. on Software Engineering and Advanced Applications, Crotia 2006.

Challagulla V.U.B., Bastani F.B., I-Ling Y., Paul R.A., *Empirical assessment of machine learning based software defect prediction techniques*, Proc. 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems 2005.

Chiu N., Huang S., *The adjusted analogy-based software effort estimation based on similarity distances*, "Journal of Systems and Software" 2007, vol. 80, no. 4.

Chulani S., Boehm B., Steece B., *Bayesian Analysis of Empirical Software Engineering Cost Models*, "IEEE Transactions on Software Engineering" 1999, vol. 25, no. 4.

Cockram T., *Gaining Confidence in Software Inspection Using a Bayesian Belief Model*, "Software Quality Journal" 2001, vol. 9, no. 1.

Collofello J.S., Yang Z., Tvedt J.D., Merrill D., Rus J., *Modelling Software Testing Processes*, Proc. IEEE 15th Int. Phoenix Conference on Computers and Communications 1995.

Barcelos Tronto I.F. de, Silva J.D. de, Sant'Anna N., *An investigation of artificial neural networks based prediction systems in software project management*, "Journal of Systems Software", 2008, vol. 81, no. 3.

Dohi T., Nishio Y., Osaki S., *Optimal software release scheduling based on artificial neural networks*, "Annals of Software Engineering" 1999, vol. 8, no. 1–4.

Fenton N., Hearty P., Neil M., Radliński Ł., *Software Project and Quality Modelling Using Bayesian Net-works*, w: *Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects*, ed. F. Meziane, S. Vadera, IGI-Global 2009.

Fenton N., Radliński Ł., Neil M., *Improved Bayesian Networks for Software Project Risk Assessment Using Dynamic Discretisation*, w: *Software Engineering Techniques: Design for Quality*, ed. K. Sacha, "IFIP International Federation for Information Processing" 2006, vol. 227.

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

Gokhale S.S., Lyu M.R., *Regression Tree Modelling for the Prediction of Software Quality*, Proc. ISSAT Int. Conf. on Reliability and Quality in Design, Anaheim 1997.

Gray A., MacDonell S., *Applications of fuzzy logic to software metric models for development effort estimation*, Proc. 1997 Annual Meeting North American Fuzzy Information Processing Society 1997.

Hearty P., Fenton N., Marquez D., Neil M., *Predicting Project Velocity in XP using a Learning Dynamic Bayesian Network Model*, "IEEE Transaction on Software Engineering" 2009, vol. 37, no. 1.

Hewett R., Kijsanayothin P., *On modeling software defect repair time*, "Empirical Software Engineering" 2009, vol. 12, no. 2.

Hochman R., Allen E., Hudepohl J., Khoshgoftaar T., *Evolutionary neural networks: A robust approach to software reliability problems*, Proc. 8th Int. Symp. on Software Reliability Engineering, IEEE Computer Society, Washington, DC 1997.

Huang S.-J., Chiu N.-H., *Optimization of analogy weights by genetic algorithm for software effort estimation*, "Information and Software Technology" 2006, vol. 48, no. 11.

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

Khoshgoftaar T., Pandya A., Lanning D., *Application of neural networks for predicting program faults*, "Annals of Software Engineering" 1995, vol. 1, no. 1.

Koten C. van, *Bayesian statistical models for predicting software development effort*, Discussion Paper 2005/08, Department of Information Science, University of Otago, Dunedin, New Zealand 2005.

Li J., Ruhe G., *Analysis of attribute weighting heuristics for analogy-based software effort estimation method AQUA+*, "Empirical Software Engineering" 2008, vol. 13, no. 1.

Mair C., Kadoda G., Lefley M., Phalp K., Schofield C., Shepperd M., Webster S., *An investigation of machine learning based prediction systems*, "Journal of Systems and Software" 2000, vol. 53, no. 1.

Pai G.I., Dugan J.B., Lateef K., *Bayesian Networks applied to Software IV&V*, Proc. 29th Annual IEEE/NASA Software Engineering Workshop 2005.

Pfahl D., Al-Emran A., Ruhe G., *A System Dynamics Simulation Model for Analyzing the Stability of Software Release Plans: Research Sections*, "Software Process: Improvement and Practice" 2007, vol. 12, no. 5.

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., *Przegląd sieci Bayesa do szacowania ryzyka w inżynierii oprogramowania*, Zeszyty Naukowe Uniwersytetu Szczecińskiego nr 476, Studia Informatica nr 21, Szczecin 2009.

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

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

Raffo D.M., Harrison W., Vandeville J., *Coordinating Models and Metrics to Manage Software Projects*, "Software Process: Improvement and Practice" 2000, vol. 5, no. 2–3.

Ramanna S., *Rough Neural Network for Software Change Prediction*, Proc. 3rd Int. Conf. on Rough Sets and Current Trends in Computing, LNCS, vol. 2475, Springer-Verlag, London 2002.

Schröter A., Zimmermann T., Zeller A., *Predicting component failures at design time*, Proc. 2006 ACM/IEEE Int. Symposium on Empirical Software Engineering, ACM, New York 2006.

Shepperd M., Kadoda G., *Comparing Software Prediction Techniques Using Simulation*, "IEEE Transactions on Software Engineering" 2001, vol. 27, no. 11.

Shepperd M., Schofield C., *Estimating Software Project Effort Using Analogies*, "IEEE Transactions on. Software Engineering" 1997, vol. 23, no. 11.

Sitte R., *Comparison of software-reliability-growth predictions: neural networks vs parametric-recalibration*, "IEEE Transactions on Reliability" 1999, vol. 48, no. 3.

Srinivasan K., Fisher D., *Machine learning approaches to estimating software development effort*, "IEEE Transactions on Software Engineering" 1995, vol. 21, no. 2.

Stefanowski J., *An empirical study of using rule induction and rough sets to software cost estimation*, "Fundamenta Informaticae" 2006, vol. 71, no. 1.

Stewart B., *Predicting project delivery rates using the Naive–Bayes classifier*, "Journal on Software Maintenance and Evolution: Research and Practice" 2002, vol. 14.

Walkerden F., Jeffery R., *An Empirical Study of Analogy-based Software Effort Estimation*, "Empirical Software Engineering" 1999, vol. 4, no. 2.

Wooff D.A., Goldstein M., Coolen F.P.A., *Bayesian Graphical Models for Software Testing*, "IEEE Trans-actions on Software Engineering" 2002, vol. 28, no. 5.

Zhang D., Tsai J.J.P., *Machine Learning and Software Engineering*, "Software Quality Journal" 2003, vol. 11, no. 2.

TECHNIQUES FOR PREDICTING DEVELOPMENT EFFORT AND SOFTWARE QUALITY IN IT PROJECTS

Summary

The most important dimensions in software project estimation are: development effort and software quality. Several predictive models have been proposed for these dimensions. Although some of these models provide useful input for decision makers, most of them are inherently limited for industrial use. The aims of this study are to: (1) compare existing applications of methods and (2) select a best technique for building intelligent and practical models for development effort and software quality prediction. In recent years various techniques were used, which are based on statistics, machine learning, artificial intelligence and similar. Authors who used a single technique often report a success their studies, only sometimes additionally noticing threats in repeatability of their predictions in other environments. Other authors compare the accuracy of predictions obtained using different techniques. The main problem in these analyses is the lack of straightforward confirmation of the usefulness of specific techniques in building predictive models. When one author finds that one technique performs the best in their study, another author obtains the best predictions using a different technique. Bayesian nets (BNs) appear to be the best suited approach to build predictive intelligent and practical model. This paper summarizes some applications of BNs in modelling different aspects of software engineering and discusses proposed Productivity Model for analysing trade-offs between effort, size and software quality.

Translated by Łukasz Radliński