Contents

1 Einleitung

In den 60iger und 70iger Jahren stieg die Größe und Komplexität von Softwareprojekten drastisch an. Softwareentwicklung wurde immer mehr kommerziell genutzt wodurch sich neue Arten von Kunden bildeten. Darunter vor allem das Militär. Durch diese Vergrößerung der Branche war des immer weniger effektiv ohne vorige Planung zu entwickeln. Als Lösung für dieses Problem entstand das Software Projekt Management(SPM), welches auch heutzutage noch eine zentrale Rolle in der Softwareentwicklung spielt.[1] Einer der Punkte mit dem sich SPM beschäftigt ist das Project cost management(PCM). Nach Bajita[1] unterteilt sich PCM in die Kategorien Software Cost Estimation und Software Effort Estimation(SCE/SEE). Aktuelle Forschung zeigt, dass genaue SCE/SEE die Chance auf ein qualitativ hochwertiges Produkt erhöht.[2][3] Jeffery et. al. beschreiben, dass SCE eines der wichtigen Werkzeuge ist um einen kompetetiven Vorteil im Bereich des IT-Service zu erhalten.[4] Aus dieser Notwendigkeit entstanden viele SCE/SEE Tools, welche Experten bei der Projektplanung unterstützen sollen.
Im Zuge dieses Kapitels werden als erstes die bestehenden SCE/SEE Techniken vorgestellt und kategorisiert. Als nächstes wird eines der beliebtesten klassischen Modelle der SCE/SEE sowie neuere Soft Computing Modelle vorgestellt und deren Vorteile und Probleme aufgelistet. In den weiteren Absätzen wird erklärt wie die Ergebnisse der Schätzungen gemessen und in Zahlen gefasst werden können, um danach auf einige Beispiele von direkten Vergleichen der Mathematischen und Soft Computing einzugehen. Zum Schluss wird anhand der vorigen Abschnitte beantwortet, ob NNs in der Softwaretechnik(SWT) eingesetzt werden können und wo deren derzeit bekannte Grenzen liegen.

COCMO

in der SEE/SCE

vorherige, vorangegangen

2 Stand der Technik

Planungsphase

In der Phase der Planung eines Softwareprojekts ist es essentiell, die Kosten und die Dauer des Projekts zu kennen. Kunden geben Aufträge und müssen wissen, wie viel welche Funktion kosten würde und wann das Projekt vorraussichtlich abgeschlossen sein wird. Grobe Schätzungen können allein durch Erfahrung der Entwickler abgegeben werden, (hier Studie einfügen wie gut man das selber schätzt) wie zum Beispiel in SCRUM, bei dem User Stories je nach Komplexität Story Points gegeben werden, welches in der Regel die Zeit und Kosten beeinflusst [citation needed]. Bei immer größer und komplexer werdenden Projekten kann die manuelle Schätzung nicht ausreichen. Dies zeigt sich im großen Zweig des SPM, welcher sich allein mit

warum

das

beschreiben den zeit und kostenfaktor

bei ansteigender komplexität …

von softwareprojekten

der Planung und Schätzungen auseinandersetzt. Eine wichtige Metrik in der SPM ist die größe der Software. Dabei kann man zwei Arten von Methoden der größenbestimmung unterscheiden. Eine der Methodenarten sind die gezählten Lines of Code (LOC) und die daraus abgeleiteten 1000 Lines of Code (KLOC). Die andere Art von Methoden sind die Function Points (FP). Anhand der Arten der Modellkonstruktion lassen sich SCE/SEE Modelle in 4 Kategorien unterteilen. Zum einen Expertensysteme, die sich auf die Erfahrung der Personen verlassen[5]. Zweitens die Linearen Systeme, die auf der Größenmessung mit FP basieren[2]. Drittens die nonlinearen Mathematischen Modelle. Viertens die neueren Soft Computing Ansätze, die entweder bisherige Methoden erweitern oder komplett neu aufbauen[6].[7]. Eines der häufigst genutzten Tools ist das Constructive Cost Estimation Model (COCOMO).[8] COCOMO lässt sich den nichtlinearen Systemen zuordnen.

eine Methodik

einer weitere Art

Messgrößen

häufigsten/beliebtesten

3 COCOMO

COCOMO wurde 1981 von Boehm entworfen um Fehler im SPM zu verringern. COCOMO ist ein algorithmisches Modell mit vielen Eingabeparametern(zahl). In COCOMO gibt es drei Typen von Projekten, gemessen an der Komplexität. Organic Mode für einfache kleine Projekte, embedded mode für innovative komplexe Projekte mit hohen Anforderungen sowie den semi-detached mode für Projekte zwischen dem organic und embedded type. Der Projekttyp wirkt sich neben den LOC signifikant auf die Endschätzung aus. Zudem nutzt das COCOMO Modell einige cost drivers(im orignialen 15 stück), welche mit Werten zwischen *sehr niedrig* und *sehr hoch* bewertet werden. Die Cost Drivers stellen dabei Gewichtungen dar, die sich ebenfalls auf den Endwert auswirken. Das Ergebnis stellt die Schätzung der Kosten in einer gewünschten Währung dar sowie die benötigten Personenmonate. Die Klassischen Modelle bringen Probleme mit sich, von denen hier ein paar beschrieben werden.

je nach art

verschieden viel

- Algorithmische Tools sind generisch und somit nicht an das spezifische Projekt angepasst.
- Daraus ergibt sich ein Mangel an bestehenden Daten für die eigene Nische. Daten von anderen Firmen sind in der Regel nicht verfügbar.
- Erfahrung in großen Projekten ist schwierig da diese eine lange Lebensdauer haben bis sie abgeschlossen werden und die Schätzung mit dem Ergebnis verglichen werden kann.
- Oft werden die Schätzungen nebenbei gemacht, nicht als eigenständigen ~~essenziellen~~ Punkt in der Planung, wodurch die Wahl der Eingabeparameter

r

negativ beeinflusst

~~vernachlässigt wird.~~
- Mangel an Flexibilität bei sich ändernden Anforderungen. Besonders im Agilen Bereich.
- Schwierigkeit und Aufwand, Projekt spezifische Daten zu sammeln, zu beurteilen und in konkreten Zahlen auszudrücken. [9][5][10]

der genannten

Einige dieser Probleme versuchen verschiedene Forscher mit Soft Computing Ansätzen zu lösen[3]

4 Soft Computing Ansätze

Unter Soft Computing versteht man vorallem Fuzzy Logic, Evolutionäre Algorithmen und Künstliche Neurale Netze (NNs). Mit ihnen kann das Problem gelöst werden, nur an ein Projekt angepasst zu sein, da sie sich leicht generalisieren lassen. Zudem sind sind sehr flexibel und anpassungsfähig, was sie bei sich ändernden Anforderungen wertvoll macht.[11] Damit sie zufridenstellende Ergebnisse liefern, müssen die Trainingsdaten passend gewählt sein.NNs mit zufällig gewählten Trainingsdaten liefern keine besseren Schätzungen als Regressionsmodelle. [12]

5 Leistungsbeurteilung

Qualität

Um die Modelle untereinander vergleichen zu können, gibt es verschiedene Methoden die Leistung/Richtigkeit zu messen. Nach Finnie et. al. ist der mean absolute relative error (MARE) die bevorzugte Fehlerbestimmungsmethode von Software-Messforschern. Der MARE wird wie folgt berechnet:

welche sich mit … beschäftigen

$$\left|\frac{estimate - actual}{estimate}\right| \tag{1}$$

estimate ist dabei der vom Modell geschätzte Wert und actual der nach der Durchführung tatsächlich gemessene Wert ist. Um herauszufinden ob das Modell über- oder unterschätzt wird der mean relative error (MRE) wie folgt berechnet:

$$\left(\frac{estimate - actual}{estimate}\right) * 100 \tag{2}$$

Ein Wert von 0 ist dabei ein perfektes Ergebnis. Je höher oder niedriger die Zahl, desto größer wurde über- oder unterschätzt.[13] Ein Ergebnis zwischen -25 und 25 gilt als gutes Ergebnis in SEE.[10]. Viele der vorgestellten Modelle werden an laufenden Industrieprojekten getestet. Das am weitesten akzeptierte Evaluationskriterium ist der MRE aus einem Satz von 63 NASA Projekten.[14]

6 Vergleich

einerseits/adererseis

Abrahamsson et. al. vergleichen in ihrem Artikel inkrementelle Ansätze mit globalen Ansätzen im Kontext der Agilen Entwicklung. Die verwendeten Modelle sind Schrittweise lineare Regression und zwei arten von NNs. Eines ihrer Ergebnisse war, dass NNs nicht generell besser sind als einfache Regression. Das Multilayer Perceptron (MLP) aus dem Test liefert zum Beispiel schlechtere Ergebnisse als die meisten Regressionsmodelle. Das Radial Basis Function (RBF) NN lieferte ähnliche Ergebnisse wie das Regressionsmodell.[10] Besser als COCOMO[14] Auch besser als COCOMO in NASA Testprojekten in über 90

7 Fazit

verschiedene

Dieses Kapitel hat die Möglichkeit aufgezeigt, wie NNs in den wichtigen Bereichen der SCE/SEE innerhalb der SWT eingesetzt werden. Wenn für gewisse Projekte passende NNs ausgewählt und korrekt konfiguriert werden, liefern NNs bessere Ergebnisse als reine Mathematische Modelle. Vorallem in der Agilen Entwicklung lösen NNs viele Probleme anderer Ansätze, ~~oft~~ auch in Verbindung und als Erweiterung der anderen Modelle. Dennoch sind sie nicht in ausschließlich allen Fällen die beste Lösung. Es existiert noch kein NN, welches eine Universallösung für alle Softwareschätzungen darstellt. Für manche Projekte bietet sich die Lösung mit NNs mehr an als für andere, vorallem da die optimale Lösung noch nicht während der Planung bekannt ist. Vorallem in Umgegungen mit wenigen Daten oder wenig Zeit sind NNs noch nicht praktikabel, da ein passendes NN zuerst entwickelt und trainiert werden muss.

insbesondere

References

[1] Bajta, M.E., Idri, A., Ros, J.N., Fernandez-Aleman, J.L., Gea, J.M.C.D., Garcıa, F., Toval, A.: Software project management approaches for global software development: a systematic mapping study. Tsinghua Science and Technology **23**(6), 690–714 (2018). DOI 10.26599/TST.2018.9010029

[2] Matson, J.E., Barrett, B.E., Mellichamp, J.M.: Software development cost estimation using function points. IEEE Transactions on Software Engineering **20**(4), 275–287 (1994). DOI 10.1109/32.277575

[3] Bilgaiyan, S., Mishra, S., Das, M.: A Review of Software Cost Estimation in Agile Software Development Using Soft Computing Techniques. In: 2016 2nd International Conference on Computational Intelligence and Networks (CINE), Computational Intelligence and Networks (CINE), 2016 2nd International Conference on, cine, p. 112. IEEE (2016). DOI 10.1109/CINE.2016.27. URL `http://search.ebscohost.com/login.aspx?direct=true{&}db=edseee{&}AN=edseee.7556814{&}lang=de{&}site=eds-live{&}authtype=shib`

[4] Jeffery, D.R., Low, G.: Calibrating estimation tools for software development. Software Engineering Journal **5**(4), 215–221 (1990). DOI 10.1049/sej.1990.0024

[5] Heemstra, F.J.: Software Cost Estimation. Handbook of Software Engineering, Hong Kong Polytechnic University **34**(10) (1992). DOI 10.1142/9789812389701_0014. URL `https://www.st.cs.uni-saarland.de/edu/empirical-se/2006/PDFs/leung.pdf`

[6] Huang, X., Ho, D., Ren, J., Capretz, L.F.: Improving the COCOMO model using a neuro-fuzzy approach. Applied Soft Computing **7**(1), 29–40 (2007). DOI 10.1016/J.ASOC.2005.06.007. URL `https://www.sciencedirect.com/science/article/pii/S1568494605000712`

[7] Huang, S.J., Lin, C.Y., Chiu, N.H.: Fuzzy decision tree approach for embedding risk assessment information into software cost estimation model. Journal of Information Science and Engineering **22**(2), 297–313 (2006)

[8] Jain, R., Sharma, V.K., Hiranwal, S.: Reduce mean magnitude relative error in software cost estimation by HOD-COCOMO algorithm. In: 2016 International Conference on Control, Instrumentation, Communication

and Computational Technologies (ICCICCT), pp. 708–712 (2016). DOI 10.1109/ICCICCT.2016.7988044

[9] Chen, Z., Menzies, T., Port, D., Boehm, B.: Feature subset selection can improve software cost estimation accuracy. ACM SIGSOFT Software Engineering Notes **30**(4), 1 (2005). DOI 10.1145/1082983.1083171. URL http://portal.acm.org/citation.cfm?doid=1082983.1083171

[10] Abrahamsson, P., Moser, R., Pedrycz, W., Sillitti, A., Succi, G.: Effort Prediction in Iterative Software Development Processes – Incremental Versus Global Prediction Models. In: First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), pp. 344–353 (2007). DOI 10.1109/ESEM.2007.16. URL https://ieeexplore.ieee.org/document/4343762

[11] Boetticher, G.D.: Using Machine Learning to Predict Project Effort: Empirical Case Studies in Data-Starved Domains. Model Based Requirements Workshop pp. 17–24 (2001). DOI 10.1.1.19.111. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.19.111{&}rep=rep1{&}type=pdf

[12] Setyawati, B.R., Sahirman, S., Creese, R.C.: Neural Networks for Cost Estimation. AACE International Transactions p. 13.1 (2002). URL http://search.ebscohost.com/login.aspx?direct=true{&}db=bth{&}AN=7197172{&}lang=de{&}site=eds-live{&}authtype=shib

[13] Finnie, G.R., Wittig, G.E.: AI tools for software development effort estimation. Software Engineering: Education and Practice, 1996. Proceedings. International Conference pp. 346–353 (1996). DOI 10.1109/SEEP.1996.534020. URL http://epublications.bond.edu.au/cgi/viewcontent.cgi?article=1006{&}context=infotech{_}pubs

[14] Khalifelu, Z.A., Gharehchopogh, F.S.: Comparison and evaluation of data mining techniques with algorithmic models in software cost estimation. Procedia Technology **1**, 65–71 (2012). DOI 10.1016/j.protcy.2012.02.013