

ANALISI DELLA RELAZIONE TRA METRICHE DI DESIGN ED EFFORT IN UN PROGETTO XP

Pekka Abrahamsson[‡], Raimund Moser[†], Alberto Sillitti[†], Giancarlo Succi[†]
[‡]VTT Electronics, Oulu, Finland

[†]Centro per l'Ingegneria del Software Applicata, Libera Università di Bolzano
pekka.abrahamsson@vtt.fi
[rmoser, asillitti, gsucci]@unibz.it

Nell'ingegneria del software ci sono solo pochi studi empirici sulla relazione tra le metriche di design ed effort associato allo sviluppo. I risultati di questi studi sono spesso contraddittori, soprattutto perchè i dati raccolti non sono ben definiti e sono di bassa qualità. Inoltre, sono rarissimi i dati empirici ricavati da progetti sviluppati con Metodi Agili. Nell'ambito dei Metodi Agili, la raccolta automatica e non invasiva di dati è assolutamente necessaria: altrimenti lo stesso metodo di sviluppo sarebbe stravolto.

In questo lavoro analizziamo la relazione tra le metriche di design di un progetto software industriale ed il relativo effort di sviluppo.

I risultati dello studio evidenziano che in questo ambito alcune metriche sono correlate con l'effort e sono buone indicatrici dell'effort di sviluppo. Questo risultato è interessante sia per i manager, sia per il singolo sviluppatore:

a) Le metriche di design possono aiutare il manager a stimare meglio i costi del progetto ed allocare le risorse.

b) Per lo sviluppatore queste metriche sono utili come indicatore per un refactoring del design e del codice.

1. Introduzione

La stima dell'effort è un tema molto importante nell'ingegneria del software: molti progetti software non sono completati e superano il budget e i tempi previsti a causa di una mancanza di una valutazione dell'effort efficace [19].

Fin dai lavori di Putnam [15], Boehm [5] e Albrecht [1] ci sono stati tanti tentativi di costruire modelli per la stima dell'effort, però, anche i risultati dei modelli più recenti sono critici, soprattutto per la mancanza di dati empirici di alta qualità. In letteratura troviamo solo un numero limitato di casi di studio, sondaggi e lavori teorici sulla relazione tra le metriche di design e l'effort di sviluppo. Ciò accade perché studi di questo tipo sono problematici per i seguenti motivi:

- In generale è molto difficile ottenere dei dati empirici da progetti industriali. Per questo motivo, molti studi usano lo stesso set di dati (per esempio i dati di COCOMO'81) per l'analisi e questo pone dei problemi seri sulla validità dei modelli ottenuti [7].
- Gli studi fatti nell'industria sono quasi sempre analisi "post-mortem", quindi, spesso, non è possibile un controllo delle variabili analizzate.
- I dati raccolti manualmente dagli sviluppatori sono spesso di qualità e granularità molto bassa. Inoltre, gli sviluppatori non considerano

importante la raccolta di metriche, anzi, spesso la vedono come tempo perso, producendo, quindi, dati inconsistenti ed errati [11].

- La raccolta manuale di metriche non é un'attività direttamente produttiva, perciò non dovrebbe essere introdotta in un processo agile [14].

Il lavoro di Alshayeb [2] é l'unico che analizza l'impatto delle metriche di design sull'effort in un ambiente agile come nell'Extreme Programming (XP). Le novità del nostro lavoro consistono in:

1. Un esperimento formale nell'industria.
2. L'utilizzo di un nuovo tool per la raccolta automatica e non invasiva delle metriche [17].
3. I dati sono di qualità e granularità alta.

I risultati ottenuti in questa ricerca sono utili ai manager, sviluppatori e clienti. Con il tool sviluppato, il manager può valutare il tempo richiesto per completare il progetto dall'inizio del progetto e, quindi, allocare le risorse correttamente. Per lo sviluppatore é importante sapere quali classi richiedono un effort maggiore per fare un refactoring delle classi problematiche. Infine, il cliente è interessato a sapere fin dall'inizio del progetto la durata dello stesso e i costi.

Studi di questo tipo sono importanti nell'ingegneria del software empirica per verificare e costruire modelli di costo.

Certamente servono numerose esecuzioni di questo studio per verificare se i risultati ottenuti sono validi per progetti simili sia nel dominio applicativo sia nel processo di sviluppo agile usato.

L'articolo è organizzato come segue. La Sezione 2 presenta una introduzione alle metriche utilizzate, la relativa letteratura e un'introduzione al problema; la Sezione 3 descrive l'esperimento e la metodologia utilizzata nello studio; la Sezione 4 presenta i risultati; la Sezione 5 descrive i limiti dello studio; infine, la Sezione 6 trae le conclusioni.

2. Stato dell'arte

2.1. Lavori precedenti

In letteratura esistono numerosi lavori sul tema della stima dei costi. Pochi però si basano su dati raccolti in progetti industriali ed analizzano l'impatto delle metriche di design sull'effort in un ambiente di sviluppo agile. Nella tabella 1 elenchiamo i lavori più recenti e legati al nostro studio.

Autori	Tipo di studio	Metriche raccolte	Processo di sviluppo	Risultati
Frakes e Succi (2001)	Studio quasi-sperimentale in un ambiente industriale usando 4 set di dati diversi	Dimensione del codice, errori, effort e riuso	Non c'è informazione sul processo di sviluppo	Un livello di riuso maggiore sembra aumentare la qualità, però non viene ottenuta nessuna informazione chiara sulla produttività.
Briand e Wüst (2001)	Esperimento formale in ambiente universitario	Dimensione del codice e metriche di coupling e cohesion; l'effort è stato raccolto manualmente	Processo di sviluppo iterativo	La dimensione del codice è un buon indicatore dell'effort, mentre altre metriche non sembrano essere importanti.
Alshayebed e Li (2003)	"Case study" in ambiente industriale usando tre progetti diversi (JDK framework e due sistemi client-server)	Metriche CK ed effort basato sulla raccolta manuale con dei "research logs"	Un processo tradizionale per il framework e due processi agili per le applicazioni client-server	Nei due processi agili è stato osservato una forte correlazione tra effort per il refactoring e error-fix verso la fine delle iterazioni, mentre nel processo tradizionale nessuna correlazione è stata trovata.
Benediktsson e Dalcher (2003)	Framework teorico senza verifica empirica	Deimensione del codice	Processi di sviluppo incrementali	Aumentando il numero di iterazioni l'effort totale diminuisce.

Tabella 1: Studi recenti sulla relazione tra metriche di design ed effort

Frakes e Succi [10] hanno analizzato l'impatto del riuso del software sulla qualità e sulla produttività. Non si è ottenuta una risposta chiara: In alcuni progetti un livello alto di riuso aumenta la produttività, in altri no.

Briand and Wüst [6] hanno studiato l'impatto di varie metriche come la dimensione del codice e metriche di coupling e cohesion sull'effort. Sembra che solo la dimensione dell'interfaccia di una classe sia importante per l'effort; metriche più complesse non sembrano essere indicatori migliori.

Alshayebed e Li [2] hanno analizzato l'impatto delle metriche object-oriented in tre progetti industriali, due hanno utilizzato un processo di sviluppo agile, l'altro un processo tradizionale. L'analisi mostra che le metriche object-oriented sono correlate con il tempo dedicato al refactoring e alla correzione degli errori nel processo agile, ma non in quello tradizionale. I dati relativi all'effort sono stati raccolti manualmente con dei *research logs* o estratti dal numero di cambiamenti del codice sorgente. La maggiore differenza col nostro studio è che nel nostro caso la raccolta dei dati è fatta tramite un tool automatico e non invasivo.

Benediktsson e Dalcher [4] hanno proposto un modello incrementale per la predizione dell'effort basato su un modello generico del tipo $E = a * Size^b$.

Secondo il fattore b , nel processo di sviluppo incrementale sembra che l'effort diminuisca in modo significativo quando il numero di incrementi aumenta. In ogni caso, gli autori non forniscono alcuna verifica empirica del modello teorico proposto.

2.2. Raccolta dati

Il software analizzato è stato sviluppato seguendo i principi del design object-oriented usando come linguaggio di programmazione Java. Per questi sistemi le metriche di Chidamber e Kemerer (metriche CK) [8] sono accettate dalla comunità come misure di prodotto valide e verificate da diversi studi precedenti [6]. Inoltre, abbiamo raccolto anche il numero di istruzioni Java (LOC) nel codice per controllare anche l'impatto del LOC sul nostro modello.

Siccome nel processo utilizzato non sono stati prodotti documenti di design, abbiamo estratto le metriche di design dal codice sorgente usando il tool PROM [17]. Inoltre, usiamo questo tool anche per raccogliere l'effort associato ad ogni classe Java.

3. Struttura della ricerca

3.1. Ipotesi

Ci poniamo due domande: prima guardiamo, se esistono delle correlazioni tra le varie metriche CK e l'effort, dopo proviamo a vedere se le metriche CK sono utili come parametri indipendenti in un modello di predizione e se è possibile trovare un modello abbastanza semplice.

Formalmente, le due domande sono formulate nel seguente modo:

- H_0 : Non esiste una correlazione tra le metriche CK e l'effort.
- H_1 : Un modello di predizione dell'effort basato sulle metriche CK non è più significativo di un modello che usa solo LOC come variabile indipendente.

Come livello di significatività usiamo 0.05, un livello generalmente accettato nell'ingegneria del software [12].

3.2. Descrizione dei dati

Il sistema studiato è un prodotto software commerciale sviluppato presso VTT ad Oulu, Finlandia. Il sistema è stato scritto in Java usando l'IDE open source Eclipse 3.0. L'obiettivo era sviluppare un sistema di monitoraggio per applicazioni mobili. Il processo di sviluppo segue in parte le pratiche dell'Extreme Programming [3]: quattro sviluppatori hanno lavorato per un totale di otto settimane usando la tecnica del pair programming. Il progetto era diviso in cinque iterazioni, la prima e l'ultima di una settimana e le altre tre di due settimane.

Il tempo lavorativo era 5 ore al giorno ed il venerdì è stato usato per rielaborare le user stories e pianificare l'iterazione successiva. L'effort totale per workstation (due sviluppatori) è stato di circa 160 ore.

Sono state applicate diverse pratiche XP come il pair programming, small releases, continuous integration, refactoring e test driven development.

Tre sviluppatori avevano una laurea di primo livello ed esperienza industriale limitata. Il quarto sviluppatore era un ingegnere del software con lunga esperienza nel settore. Il team ha lavorato insieme in un grande ufficio e, prima dell'inizio del progetto, è stata fatta una introduzione e training sulle pratiche XP, in particolare sulla pratica del test first. Durante il progetto, un team ha monitorato e supportato l'andamento del progetto per fornire training sulle pratiche XP e aiuto in caso di problemi.

Siccome per la raccolta dei dati abbiamo usato un tool non invasivo, non c'è stato pericolo che l'introduzione del tool abbia cambiato le abitudini degli sviluppatori ed il processo di sviluppo.

Il sistema analizzato è diviso in trenta classi per un totale di 1776 LOC. L'effort totale per due sviluppatori lavorando in coppia è stato 160 ore: il 75% del tempo è stato usato per la codifica (in Eclipse) e il 25% per altri lavori (scrivere/leggere documenti, internet, email).

La tabella 2 presenta un riassunto delle metriche CK e dell'effort per le 30 classi.

	Minimo	Massimo	Media	Deviazione std.
EFFORT (minuti)	4	1181	130	213
LOC	6	378	59	83
CBO	1	32	10	7
DIT	1	4	2	1
NOC	0	2	0	0
LCOM	0	124	8	26
RFC	0	91	26	20
WMC	0	58	14	14

Tabella 2: Statistica descrittiva delle metriche

3.3. Procedura di analisi dei dati

Il nostro primo obiettivo è quello di determinare se le varie metriche CK sono correlate tra loro e con l'effort. Come misura di correlazione usiamo il coefficiente di correlazione di Spearman, perchè non dipende dalla popolazione sottostante ed è più stabile nella presenza di outlier [16]. Questa prima analisi è utile per trovare delle collinearità tra le varie metriche che potrebbero causare problemi in un modello di predizione e per avere un primo indicatore della relazione tra le metriche CK e l'effort.

In secondo luogo vogliamo capire quali delle metriche CK sono opportune variabili indipendenti in un modello di stima dell'effort e se le metriche CK aggiungono un ulteriore valore nella stima dell'effort in confronto alle sole LOC. Usiamo il metodo della *stepwise linear regression* [13] per questo scopo.

4. Analisi dei risultati

4.1. Correlazione

La figura 1 presenta un box-plot delle metriche CK che abbiamo estratto dal codice sorgente. Si vede che ci sono alcuni outliers con dei valori abbastanza grandi.

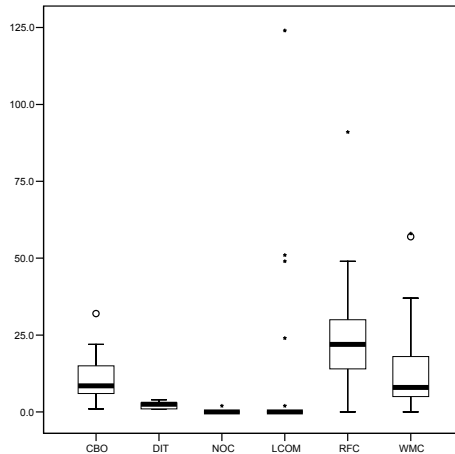


Figura 1: Box-plot delle metriche CK

Siccome nel nostro caso – visto anche che il nostro sample è abbastanza piccolo – non è ragionevole di escludere outlier dall’analisi, dobbiamo essere particolarmente attenti ai risultati ottenuti dalla regressione lineare, essendo molto sensibile alla presenza di outliers.

Il coefficiente di correlazione di Spearman invece si basa sul rango dei valori assoluti delle metriche CK, perciò non è particolarmente sensibile alle singolarità.

La tabella 3 presenta il risultato del calcolo del coefficiente di Spearman per le metriche di design:

		CBO	DIT	NOC	LCOM	RFC	WMC
Spearman's rho	CBO	1	0.54(**)	-0.28	-0.26	0.74(**)	0.24
	DIT	0.54(**)	1	-0.20	-0.28	0.34	-0.03
	NOC	-0.28	-0.20	1	0.45	0.0	0.21
	LCOM	-0.27	-0.28	0.45	1	0.21	0.54(**)
	RFC	0.74(**)	0.34	0.00	0.21	1	0.68(**)
	WMC	0.24	-0.03	0.21	0.54(**)	0.68(**)	1
	**Correlazione significativa al 0.01						

Tabella 3: Analisi di correlazione delle metriche CK

Dalla tabella 3 vediamo che diverse metriche CK sono correlate tra loro: i valori delle metriche RFC e CBO sono correlati perchè entrambi sono misure di coupling. La metrica RFC è anche correlata con WMC: anche questo fatto è una conseguenza della loro definizione. Una correlazione più interessante è

quella tra LCOM e WMC: se aumenta il numero di metodi pesati con la loro complessità ciclomatica, anche il valore di LCOM sembra aumentare. Questo fatto potrebbe indicare che diventando più complesse le classi tendono a perdere coesione: in questi casi sembra opportuno dividere la classe in due o più classi che offrono la stessa funzionalità ma in modo più coeso.

Tutto sommato possiamo dire che le metriche di design sono correlate tra loro e che è probabile che siano anche collineari. Di questo fatto dobbiamo tenerne conto quando usiamo le metriche CK come variabili indipendenti in modelli di regressione [18].

La tabella 4 ci mostra le correlazioni significative tra le metriche CK e l'effort.

	CBO	RFC	WMC
Effort per classe	0.65**	0.78**	0.53**

**Correlazione significativa al 0.01

Tabella 4: Correlazione tra effort e metriche di design

RFC è molto correlato con l'effort; lo stesso vale anche per CBO e WMC.

Perciò possiamo rifiutare l'ipotesi H_0 : le metriche di design, in particolare le metriche di coupling (CBO, RFC) e complessità (WMC) sono correlate in modo significativo con l'effort.

4.2. Stepwise linear regression

Costruiamo un modello lineare per l'effort, o una sua trasformazione [20], come variabile dipendente e, come variabili indipendenti, consideriamo le metriche CK.

A tal fine usiamo il metodo della *stepwise linear regression*. Per trovare un modello semplice seguiamo il principio di parsimonia: se un modello complesso spiega un fatto solo un poco meglio di un modello semplice, si deve usare il modello semplice.

Il nostro primo tentativo consiste nel costruire un modello del tipo:

$$Effort = \sum_i a_i CK_i, CK_i \in \{CBO, DIT, NOC, LCOM, RFC, WMC\} \quad (1)$$

Questo modello ci da un risultato abbastanza buono (è in grado di spiegare ca. 65% della variazione nell'effort), pero non soddisfa le quattro condizioni necessarie per poter costruire un modello lineare valido: distribuzione normale, omoschedasticità, indipendenza, linearità.

Se queste condizioni non sono rispettate, il modello ottenuto è critico nel senso che la varianza dell'errore può essere alta. Una trasformazione del tipo $Effort \rightarrow Effort^{0.5}$ cambia la distribuzione dell'effort in modo tale che le quattro condizioni sono rispettate. Per confrontare i modelli basati sulle metriche CK e LOC abbiamo prima sempre applicato una trasformazione di questo tipo sulla variabile effort.

Siccome le LOC sono altamente correlate con l'effort ed alcune metriche di design, la prima idea è che forse le LOC abbiano il maggior impatto sull'effort in un modello di predizione. Studi precedenti sembrano confermare questo

sospetto [6]. Per determinare se questo sia vero anche nel nostro caso confrontiamo il nostro modello originale con altri due :

- Un modello che aggiunge LOC come variabile indipendente al nostro modello
- Un modello che usa solo LOC come variabile indipendente

Confrontiamo i tre modelli usando dei parametri ben noti nei modelli di costo [9]: Mean Magnitude of Relative Error (MMRE), la sua media (MdMRE) e la predizione di livello k (PRED(k)). PRED(k) è il numero di osservazioni dove il MRE è minore di k .

La tabella 5 ci illustra i risultati di questo calcolo.

Modello	Tutte le metriche CK	Tutte le metriche CK + LOC	Solo LOC
Predittori	CBO, WMC	CBO, WMC, LOC	LOC
R^2	0.72	0.74	0.47
MMRE	0.37	0.35	0.48
MdMRE	0.26	0.27	0.29
PRED (0.25)	50%	46%	36%

Tabela 5: Confronto dei tre modelli di predizione dell'effort

Dalla tabella 5 vediamo che il nostro modello ed il modello che include tutte le metriche CK e LOC possiedono prestazioni paragonabili: Siccome possiamo raccogliere le metriche di design nella fase iniziale dello sviluppo, mentre LOC è solo conosciuto alla fine dello sviluppo, il nostro modello basato sulle metriche CK è chiaramente migliore. Il modello che usa solo LOC come variabile indipendente, invece, è molto semplice ma inferiore al nostro modello.

Tutto sommato possiamo rifiutare la nostra ipotesi H_1 : Un modello di predizione per l'effort che si basa sulle metriche CK come variabili indipendenti è chiaramente superiore al modello che si basa solo sulle LOC.

5. Limiti dello studio

Non abbiamo conoscenza di altri studi di questo tipo e non possiamo confrontare i nostri risultati con quelli di studi indipendenti. Pensiamo che in futuro sia possibile replicare l'esperimento in vari siti e che potremo verificare o meno i risultati qui presentati.

5.1. Generalizzazione dei risultati

Siccome la dimensione del nostro campione è abbastanza piccola ed abbiamo analizzato solo un progetto particolare sviluppato usando pratiche XP nel dominio delle applicazioni mobili, risulta molto difficile trarre delle conclusioni generali. Abbiamo visto che diverse metriche di design sono correlate all'effort di sviluppo e sono utili per costruire un modello di predizione, ma questo fatto non implica una relazione causale. Esperimenti futuri dovranno evidenziare se i risultati ottenuti possono essere generalizzati per progetti sviluppati in ambienti agili e per applicazioni mobili o no.

6. Conclusioni

Questa ricerca presenta un'analisi empirica della relazione tra metriche di design ed effort di sviluppo in un progetto software industriale sviluppato con XP. Abbiamo potuto fare un esperimento formale usando un tool innovativo per la raccolta delle metriche analizzate che ci ha permesso di avere dati molto dettagliati e accurati.

Le implicazioni di questa ricerca sono:

- Alcune delle metriche di CK sono altamente correlate, perciò si devono considerare effetti di collinearità quando usati in un modello di predizione.
- Le metriche di coupling e complessità sono correlate in modo significativo con l'effort di sviluppo.

Per uno sviluppatore questo risultato è particolarmente interessante, perchè sembra essere meglio dividere una classe con valori alti di coupling e complessità in due o più classi con un CBO e WMC più basso per diminuire l'effort di sviluppo.

Futuri esperimenti formali sia in ambienti simili sia diversi potranno confermare o meno i nostri risultati. In particolare, sarebbe interessante raccogliere dati relativi ai difetti. In questo caso, potremmo analizzare non solo la relazione tra le metriche CK e l'effort, ma anche l'impatto dell'effort e delle metriche di design sulla qualità del prodotto finale.

Ringraziamenti

Ringraziamo in particolare il team di sviluppo di VTT, ad Oulu, che si è mostrato disposto ad installare il nostro tool per la raccolta dati. Gli autori ringraziano anche il Ministero dell'Istruzione, dell'Università e della Ricerca per il progetto FIRB MAPS (<http://www.agilexp.org>) e la Provincia Autonoma di Bolzano per il progetto INTERREG Software District (<http://www.caso-synergies.org>).

7. Riferimenti bibliografici

- [1] Albrecht A. J., Gaffney J. E., 1983 "Software function, source lines of code, and development effort prediction," *IEEE Transactions on Software Engineering*, **9**(6): 639-648
- [2] Alshayeb, M. and Li, W., 2003, "An Empirical Validation of Object-Oriented Metrics in Two Different Iterative Software Processes," *IEEE Transactions on Software Engineering*, **29**(11): 1043-1048, November
- [3] Beck, K., 2000, *Extreme Programming Explained: Embrace Change*, Addison-Wesley
- [4] Benediktsson, O. and Dalcher, D., 2003, "Effort estimation in incremental software development," *IEEE Proceedings Software* **15**(6): 351:357, December
- [5] Boehm, B., 1981, *Software Engineering Economics*. Englewood Cliffs, NJ Prentice-Hall
- [6] Briand, L., Wüst J., 2001, "Modeling Development Effort in Object-Oriented Systems Using Design Properties," *IEEE Transactions on Software Engineering*, **27**(11): 963-986, November
- [7] Campbell, D., T., Stanley, J., C., 1966, *EXPERIMENTAL AND QUASI-EXPERIMENTAL DESIGNS FOR RESEARCH*. Houghton Mifflin Company Boston
- [8] Chidamber, S., Kemerer C., F., 1994, "A metrics suite for object-oriented design," *IEEE Transactions on Software Engineering*, **20**(6): 476-493, June
- [9] Conte, S., D., Dunsmore, H., E., Shen, V., Y., 1986, *Software engineering metrics and models*, The Benjamin/Cummings Publishing Company, Inc.
- [10] Frakes W. B., Succi G., 2001, "An Industrial Study of Reuse, Quality, and Productivity," *Journal of Systems and Software*, **57**(2):99-106

- [11] Johnson, P., M., Kou, H., Agustin, J., M., Zhang, Q., Kagawa, A., Yamashita, T., 2004, "Practical automated process and product metric collection and analysis in a classroom setting: Lessons learned from Hackystat-UH," *Proceedings of the 2004 International Symposium on Empirical Software Engineering (ISESE '04)*, Redondo Beach, California
- [12] Mišić, V., B. and Tešić D., N., 1997, "Estimation of effort and complexity: An object-oriented case study," *The Journal of Systems and Software*, **41**(2): 133-143
- [13] Myers, J., L. and Well, A., D., 2003, *Research Design and Statistical Analysis*, Lawrence Erlbaum Associates Publishers, Mahwah, New Jersey
- [14] Poppendieck T., Poppendieck M., 2003, *Lean Software Development: An Agile Toolkit for Software Development Managers*, Addison-Wesley
- [15] Putnam, L., H., A., 1978, "A general empirical solution to the macro software sizing and estimation problem," *IEEE Transactions on Software Engineering*, **4**(4): 345-381, July
- [16] Siegel, S. and Castellan, N.J., 1988, "Nonparametric Statistics for the Behavioral Sciences," McGraw-Hill, ch. 9.3, ch. 6
- [17] Sillitti, A., Janes, A., Succi, G., Vernazza, T., 2003, "Collecting, Integrating and Analyzing Software Metrics and Personal Software Process Data," *Proceedings of the EUROMICRO 2003*, p. 236
- [18] Succi, G., Pedrycz, W., Djokic, S., Zuliani, P., and Russo, B., 2005, "An Empirical Exploration of the Distributions of the Chidamber and Kemerer Object-Oriented Metrics Suite," *Empirical Software Engineering* **10**(1): 81-104, Kluwer Academic Publisher
- [19] The Standish Group, 1994, "The CHAOS report (2004)," http://www.standishgroup.com/sample_research/PDFpages/q3-spotlight.pdf
- [20] Tukey, W., J., 1977, *Exploratory Data Analysis*, Addison-Wesley