IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP5008

Can Business Process Changes be Cheaper Implemented with Workflow Management Systems?

Niko Kleiner
Department for Programming Methodology and Compiler Construction , Faculty of Computer Science , 89069 University of Ulm ,
email: nikolaus.kleiner@informatik.uni-ulm.de

ABSTRACT

Workflow Management Technology has received broad attention over the last decade. Despite their publicity, their value - in terms of money - has been subject to assumptions and speculations. This paper reports about an experiment that compares the efforts needed to implement business process changes with Lotus Domino Workflow on the one hand and Lotus Domino Designer on the other. It describes the threats to the validity of the experimental results, possible mitigations, the experiment design, and the statistics used for data analysis. Of course, the experiment needs to be replicated to obtain valid data. First results show a trend of 30% savings of total implementation effort when using the Workflow Management System and even 30% up to 90% with re-spect to pure process implementation efforts.

1. MOTIVATION

Today, the introduction or reengineering of business processes usually comes along with the development of a supporting Information System. Rapidly changing envi-ronments, learning and a turbulent market force any company to change their busi-ness processes frequently [1,11]. A critical challenge for any enterprise is to be able *to react to those changes quickly and effectively* [2,10]. Most process changes also have to be propagated to the supporting PaIS. The IS community has responded with Workflow Management Systems (WfMS). WfMS claim to make changes to the implemented workflow easier (the term workflow refers to the formalized, com-puter supported part of the business process).

Different solutions are available on the market, for example, IBM MQ Series, Vitria BusinessWare, BEA Systems, Staffware and Lotus Workflow. The core idea of a WfMS is to separate the implementation of the workflow logic from the rest of the code. The workflow is described as an explicit, mostly graphical definition that is interpreted during runtime by a so called workflow engine. This approach is straightforward. *But what is the value of WfMS – in terms of money saved?* Our paper picks up this question.

The contributions of our paper are as follows: first, we give a detailed descrip-tion of a specific experimental design to address the question, including a discussion of the specific threats that need to be taken into account (section 2). So, the experi-ment can be replicated. Second, we present a data analysis framework (section 3) and third, we report about our first results (section 4). Of course, the experiment must be replicated to obtain generalizable data (cf. [4]). However, the data already show a trend.

2. EXPERIMENT DESIGN AND CONDUCTION

The literature about Software Engineering experimentation [3,8,15,18] describes various designs and guidelines to set up an appropriate experiment. Yin's remarks about evaluation criteria for empirical social research [17] and Murphy's study about appropriate experimental methods for emerging development techniques [14] also contributed to our eventual design. A discussion of the different possibilities is beyond the scope of this paper. Our (rough) design is common to compare two software development aids (randomised paired comparison, cf. [8]). Regarding the experiment design, the contribution of this paper is the detection, discussion and mitigation of the threats *specific* to this kind of experiment.

We first discuss the threats to internal and external validity. Then we describe the experiment design and conduction.

1.1 Threats to Internal Validity

People: Software developers strongly differ in their skills and thus, in their produc-tivity. We consider the following reasons: general experience with software devel-opment and experience with the software development environment.

The first problem cannot be solved in general except conducting the experiment with a sufficiently large and representative set of subjects. We mitigated it by choos-ing students from the same year with almost the same software development experi-ence (years of experience). In our experiment we focus on process *changes*. Therefore, we provided a "base implementation". The students were asked to implement process changes. This also mitigates the problem since more experienced programmers usually design their software better and hence, process changes might be implemented easier due to a "clever" design. A base implementation reduces such design choices.

We eliminated the second threat by picking a development environment that was unknown to every participant (Lotus Domino).

Effort Data Collection Process: One needs to ensure that the participants pre-cisely understand *what data* is to be collected. Data collection is one of the most critical aspects and also needs to be controlled carefully while executing the ex-periment.

We used a simple effort classification: effort spent to implement database func-tionality, process functionality and effort spent for other activities (e.g., testing, debugging, reading of documentation etc.). We prepared effort data collection forms. Before starting the real experiment, we taught data collection in a pilot pro-ject. Further, we had weekly meetings to review and discuss the collected data. In earlier meetings data sometimes had to be reclassified due to misunderstandings. This phenomenon disappeared quickly.

Extent of Implemented Functionality: This threat is also related to data collec-tion: Some developers might spent more effort for a "nice" user interface, for exam-ple, than others do. To minimize such effects, we worked out detailed requirements specifications. Every week, the implemented parts were reviewed with respect to this specification and, if necessary, the collected data was adjusted.

1.2 Threats to External Validity

Using Students instead of Professionals: Runeson already discussed this topic in detail [16]. With respect to his studies, absolute values are difficult to generalize. Findings about trends seem to be transferable. Houdek reports about similar results in his work about the value of university experiments to improve industrial software development processes [7]. Thus, we only analyze relative effort savings.

Choice of Implementation Technologies: The technologies used in the experi-ment must be representative and comparable.

We chose Lotus Domino Designer and Workflow since Lotus Notes is the stan-dard communication platform at our industry partner's site. It has already been used in several projects to support business processes. Researchers that replicate the experiment should use Lotus Domino (to validate our results) as well as other de-velopment environment pairings (to contribute to more general results).

Not every implementation technology can be compared to another. To get rea-sonable results, the candidate technologies should at least be used in current practice to implement business process support. For our first trials we wanted technologies that are as similar as possible to maximize the influence of the WfMS architecture. Consequently, other benefits of the technology would not affect our results so much. Lotus Domino Designer and Workflow fulfill these requirements.

Choice of Examples: By accident, we just could have chosen appropriate busi-ness process projects that strongly support our hypothesis. We picked examples from two different departments to mitigate this effect.

1.3 Experiment Preparation

The experiment preparation phase started in June and lasted until October (see fig-ure 1). In a pilot project, three teaching assistants (TA) implemented several exam-ples with Domino Designer and Workflow. We also tried different data collection techniques. In parallel, we collected project data from our industry partner.

Out of this data we prepared an introduction into the problem domain (car de-velopment) and four specifications (PSS_1, PSS_2, SCM_1, SCM_2) for two different business process support projects (PSS = Packaging Support System, SCM = Super Change Management). Each such specification describes process changes that the students were supposed to implement in the future experiment.

Based on the introductory documentation the teaching assistants built a base im-plementation. The students used this implementation during the experiment to add their changes.

We also used a month to train the students in the new problem domain and in Domino Designer and Workflow.

1.4 Experiment Design and Conduction

As you can see in figure 1, the actual experiment started mid-October and lasted until the end of February. Four students from the fourth and fifth year participated.

Each student implemented his/her own system, two of them using Domino De-signer and two of them using Domino Workflow. All changes were added to the given base implementation. Every week, they implemented some changes and the students recorded their efforts spent. In average, the students spent ten hours a week for reading, debugging and implementation. Every week, a meeting was held to review the resulting systems with respect to the specification. The results, the problems and the collected data were extensively discussed.

With respect to the weekly increments the students deviated in progress. We synchronized them with respect to every major increment (compare figure 1) such that at the end of every such increment there were four different implementations of the same specification. Figure 2 is a screenshot of a resulting SCM_1 Designer im-plementation.

We could have switched the development environments and teams in the middle of the experiment. This would have further improved the quality of our data but would have needed additional training. In favour of more time for the experiment we did not switch the environments.

Figure 1: Actual Project Plan, Including Preparation and Training Phase

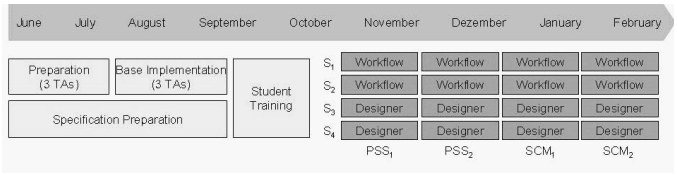
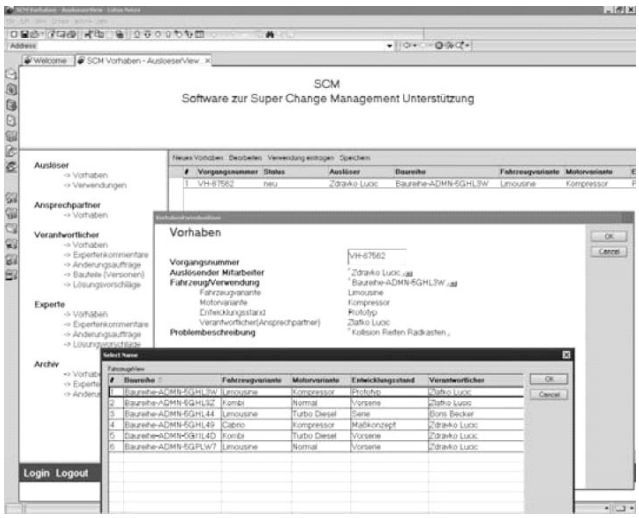


Figure 2: Screenshot of a SCM1 Designer Implementation



3. DATA ANALYSIS FRAMEWORK

There are three independent variables: development environment (Domino Work-flow, Domino Designer), student (S_1, S_2, S_3, S_4) and requirements set (PSS_1, PSS_2, SCM_1, SCM_2). The variable of interest is the development environment. The de-pendent variable is the time spent for implementation with respect to fixed values for the independent variables. Hence, we get sixteen independent measurements. We had to accumulate the time spent for implementation for every major increment (compare the section about the experiment conduction).

We are interested in the effort saved when using Domino Workflow instead of Domino Designer. Within every increment the effort data is comparable and inde-pendent from each other. Cross-increment comparison can only be made when con-sidering percentages, since the requirements sets cannot be guaranteed to be of equal size. Further, we need to take care that the comparisons again yield independ-ent measures. This is a necessary condition to be able to apply statistical methods. Thus, there are two possibilities for comparison: comparing S_1 with S_3 and S_2 with S_4 (in the following called S_1/S_3-S_2/S_4 pairing) or comparing S_1 with S_4 and S_2 with S_3 (in the following called S_1/S_4-S_2/S_3 pairing). Note that a comparison of S_1 and S_3 and S_1 and S_4 would *not* yield independent measures. Another threat to independ-ence are the requirements sets. It could be that the measures obtained from imple-menting PSS_1 are not independent from those obtained from PSS_2, since PSS_2 might depend on PSS_1. We mitigate this, since we provide a base implementation and only let the students implement changes. We observed that we also could have started with PSS_2 and then have proceeded with PSS_1. The same holds for SCM_1 and SCM_2.

For the S_1/S_3-S_2/S_4 pairing, we define the random variable X_i as the percentage of effort saved when implementing with Domino Workflow compared with Domino Designer, i=1,…,8. X_1 are the savings of S_1 compared with S_2 with respect to PSS_1, X_2 the savings of S_2 compared with S_4, X_3 the savings of S_1 compared with S_3 with respect to PSS_2 and so on. Y_i is defined analogously for the S_1/S_4-S_2/S_3 pairing, i=1,…,8. X_i and Y_i are independent and equally distributed random variables. Our sample size is eight. Note that we do not know their kind of distribution. The data analysis framework follows:

1. Compute the sample mean, sample standard deviation and the median.
2. Apply the *Kolmogoroff/Smirnov* test to check if the sample indicates normal distribution of the underlying universe.
3. Check, if the mean of the workflow-sample is significantly less than the mean of the Designer-sample. We can do this by testing if the mean of our random variable is significantly greater than zero. Hence, the 0-hypothesis should be formulated as "X (or Y) has a mean of less than or equal to zero".

If normal distribution can be assumed, apply the *one sided t-test for paired samples*. In addition, one can calculate a *confidence interval* for the mean. Note that we are interested *how much* we save in average.

If Kolmogoroff/Smirnov fails, apply the *one sided Wilcoxon rang-sum test for paired samples*. In this case, however, no confidence interval can be computed.

The tests are described in the literature in more detail (cf. [5,9]).

4. DATA ANALYSIS AND INTERPRETATION

Table 1 and 2 list our results. X_i^{db} denotes effort data for database implementation and X_i^p for process implementation. Otherwise, they are defined like X_i. Y_i^{db} and Y_i^p are defined analogously. For example, X_4=83, X_4^{db}=-50 and X_4^p=93 means that S_2 needed 83% less time than S_4 to implement PSS_2. If we restrict our considerations to the time spent to implement database functionality he needed 50% more time and, if we only consider process functionality, 93% less time.

The *KS*-column lists the results of the Kolmogoroff/Smirnov test statistic. The critical value on a 5%-level of significance is 0.454. All results are greater than the critical value, and hence the 0-hypothesis "The sample indicates no normal distribu-tion" cannot be rejected. So we cannot assume normal distribution. This is not what we expected. Experiment replications have to show if this is the usual case or not.

Therefore, the results of the t-test as well as the calculation of the confidence in-tervals is not valid. The corresponding columns are shadowed in table 1 and 2 and only serve as an example. We demonstrate the analysis methods anyway, since this would be the preferred method. For this experiment execution, only the results of the Wilcoxon test and the calculations of the sample mean, sample standard devia-tion and median are valid.

The exemplary results of the t-test are listed in table 1 and 2 in column *t*. The critical value on a 5%-level of significance would be 1.89. The test statistic is greater than the critical value for the X_i-, the X_i^p- and Y_i^p-sample. Thus, the 0-hypothesis "The usage of the workflow environment yields no significant saving in implementation effort" could have been rejected in these cases.

In table 1 and 2, the calculations of the confidence intervals (for the correspond-ing means) are listed in column *C*. In our case, the confidence intervals and the results of the t-test are related: the confidence interval is in a fully positive range iff we can reject the 0-hypothesis. This fact can be used to validate the calculations.

The confidence interval calculations listed in table 1 would have told us that there would have been a saving in total implementation efforts of at least 27% and up to 70% with a probability of 95% (again, we use a 5%-level of significance). The implementation effort for database functionality would have ranged from 64.2% more effort to 43.4% less. If we only consider the effort to implement process functionality, we would have saved at least 66.6% and up to 88.2%.

Even though we cannot assume normal distribution, the Wilcoxon rang-sum test can be applied. It is a weakening of the t-test. The test statistics are listed in column *W* in table 1 and 2. The critical value is 4, except for the Y_i^{db}-sample, where it is 2. We can reject the 0-hypothesis (same as for the t-test) if the test statistic is less than or equal to the critical value. We obtain the same results as for the exemplary t-test. So we can say that for the S_1/S_3-S_2/S_4 pairing there was a significant saving in total and process implementation effort and for the S_1/S_4-S_2/S_3 pairing only for the latter. Thus, we can conclude - on a 5%-level of significance - that there is a significant saving for process implementation. The data indicates that this also results in a sav-ing of total implementation effort, although we cannot conclude it statistically. But a look at the data suggests that this problem seems to come from some extreme out-liers in the S_1/S_4-S_2/S_3 pairing.

Whatever distribution we have, we can analyze the sample mean ¯#µ, the sample standard deviation ¯#ó and the median *m*. ¯#µ indicates a saving in total efforts for both pairings (48.5% and 28.8%), more effort for database implementation (-10.4% and –52.3%), and high savings for process implementation (77.4% and 69.4%). The high sample standard deviations for the X_i^{db}-, the Y_i- and Y_i^{db}-sample are problem-atic. One can see that the tests failed exactly for these cases. Thus, we can conclude that we can expect a lower limit of about 33% (=69.4-36.5) for the effort savings with respect to process functionality and an upper limit of 90.3% (77.4+12.9). ¯#ó = 85.8 for the Y_i-sample makes it difficult to say something about the total effort saved. As already mentioned it looks like this is due to some high outliers. For ex-ample, the median – which is more robust against outliers than the sample mean – indicates that we can expect savings of at least 43% in total, around 0% for database functionality and at least 79% for process functionality.

Table 1: Results with Respect to the S1/S3-S2/S4 Pairing

i	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*KS (0.45)*	$\bar{\mu}$	$\bar{\sigma}$	*m*	*W (4)*	*t (1.89)*	*c*
X_i	31	79	41	83	65	37	45	7	0.48	48.5	25.7	43	0	5.3	27 70
X_i^{db}	4	66	-67	-50	65	-3	20	-118	0.49	-10.4	64.4	0.5	16	-0.5	-64.2 43.4
X_i^p	79	94	66	93	62	84	62	79	0.46	77.4	12.9	79	0	16.9	66.6 88.2

Table 2: Results with Respect to the S1/S4-S2/S3 Pairing

i	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*KS (0.45)*	$\bar{\mu}$	$\bar{\sigma}$	*m*	*W (4)*	*t (1.89)*	*c*
Y_i	36	78	66	70	81	-14	81	-168	0.50	28.8	85.8	68	9	0.9	-42.9 100.5
Y_i^{db}	-6	70	-150	0	69	-15	68	-454	0.50	-52.3	177.8	-3	11 (2)	-0.8	-201 96.3
Y_i^p	85	92	83	86	94	-8	88	35	0.49	69.4	36.5	85.5	1	5.4	38.9 99.9

5. RELATED WORK

Korson and Vaishnavi [12] investigated the benefits to maintenance of using modu-lar code against non-modular. The results determined a significant difference in favour of the modular code. Daly et al. [4] replicated the experiment. They could not confirm the results and criticized that the modular programs contained com-ments the monolithic one did not and thus favored the modular one. Second, they argued that the activities in the experiment to modify the code did not correspond to a "normal" work process. In our experiment we paid attention to these points.

Misra and Jalics [13] compared third- (COBOL) and fourth-generation (sBaseIII, PC-Focus) programming languages with respect to development effort, code size, and performance characteristics. They found that forth-generation lan-guages do not shorten development time per se but COBOL always yielded faster code. We did not include such performance measurements into our experimental setting.

Harrison et al. [6] implemented a set of image processing algorithms with a functional programming language (SML) on the one hand and an object-oriented (C++) on the other. They also measured development time and did not find a sig-nificant difference.

6. SUMMARY AND OUTLOOK

In summary we can conclude the following:

- The effort savings to implement database functionality alternates, but is close to zero in average
- The trend with respect to process implementation ranges from at least about 30% up to about 90%
- The trend for total savings lies at about 30%

The experiment must be replicated to obtain generalizable data. If, in the long term, we can assume normal distribution for most cases, the calculation of the confidence intervals is the preferred way. Otherwise, we can only apply the Wilcoxon test.

In our future work we also include graphical analysis methods, for example, main effect plots and normal plots (cf. [9]). The results also depend on the require-ments that need to be implemented. Our results show that if there were no process functionality than the use of Domino Workflow would not be an advantage. Thus, it would be interesting to know how we can estimate this influence from a given specification.

ACKNOWLEDGMENTS

Many thanks to Stefan Sarstedt, Jens Kohlmeyer and Matthias Schneiderhahn for their indispensable support during the experiment. I'm in particular thankful to Joachim Herbst and Jens Staudt for their support in gathering project data for the specifications.

LITERATURE

1. Agostini, A., De Michelis, G.: Improving Flexibility of Workflow Management Systems. In: Proceedings of the International Conference on Business Process Management (BPM'00), LNCS 1806, 218-234, 2000

2. Van der Aalst, W.M.P., Jablonski, S. (eds.): Flexible Workflow Technology Driving the Net-worked Economy. International Journal of Computer Systems, Science and Engineering, 15(5), 267-276, 2000

3. Basili, V.R., Selby, R.W., Hutchens, D.H.: Experimentation in Software Engineering. IEEE Transactions in Software Engineering, SE-12(7), 733-743, 1986

4. Daly, J., Brooks, A., Miller, J., Roper, M., Wood, M.: An External Replication of a Korson Ex-periment. Technical Report Empirical Foundations of Computer Science EfoCS-4-94, Univer-sity of Strathclyde, Glasgow, 1994

5. Graff, J. Nichtparametrische Statistik in den Sozialwissenschaften (in german). Centataurus-Verlagsgesellschaft, 1998

6. Harrison, R., Samaraweera, L.G., Dobie, M.R., Lewis, P.H.: An Empirical Evaluation of Func-tional and Object-Oriented Languages. Research Journal, Department of Electronics and Computer Science, University of Southampton, 133-135, 1994

7. Houdek, F. Empirisch basierte Qualitätsverbesserung – Systematischer Einsatz externer Experi-mente im Software Engineering (in german). Logos-Verlag, Berlin, 1999

8. Juristo, N. and Moreno, A.M.: Basics of Software Engineering Experimentation. Kluwer Aca-demic Publishers, Boston, 2001

9. Kenett, R., Zacks, S.: Modern Industrial Statistics: Design and Control of Quality and Reliabil-ity. Brooks/Cole Publishing Company, 1998

10. Klein, M., Dellarocas, C., Bernstein, A.: Special Issue on Adaptive Workflow Systems. Interna-tional Journal of Collaborative Computing, 9(3-4), 2000

11. Kleiner, N.: The Focus of Requirements Engineering in Workflow Application Development. In: CaiSE'03 Workshop Proceedings, Workshop on Requirements Engineering for Business Process Support (REBPS'03), 372-378, RWTH Aachen, 2003

12. Korson, T.D., Vaishnavi, V.K.: An Empirical Study of the Effects of Modularity on Programm Modifiability. In: E. Soloway and S. Iyengar (eds.): Empirical Studies of Programmers: First Workshop, 168-186, Ablex Publishing Corporation, 1986

13. Misra, S.K., Jalics, P.J.: Third-Generation versus Fourth-Generation Software Development. IEEE Software, 5(4), 1988

14. Murphy, G.C., Walker, R.J., Baniassad, E.L.A.: Evaluating Emerging Software Development Technologies: Lessons Learned from Assessing Aspect-oriented Programming. IEEE Transac-tions on Software Engineering, 25(4), 1999

15. Pfleeger, S.L.: Design and Analysis in Software Engineering, Part 1-4. ACM SIGSOFT Software Engineering Notes, 19(4)-20(3), 1994-1995

16. Runeson, P.: Using Students as Experiment Subjects – An Analysis on Graduate and Freshmen Student Data. In: Proceedings of the 7th International Conference on Empirical Assessment & Evaluation in Software Engineering (EASE'03), 2003

17. Yin, R.K.: Case Study Research: Design and Methods (2nd Edition). Sage Publications, Thou-sand Oaks, CA, 1994

18. Zelkowitz, M.V., Wallace, D.R.: Experimental Models for Validating Technology. Computer, 31(5), 23-31, 1998