Original Research Paper

Volume : 5 | Issue : 7 | July 2016 • ISSN No 2277 - 8179 | IF : 3.508 | IC Value : 69.48

Metric Based Evaluation of Software Paradigms

Engineering

KEYWORDS :

V. Rajiv Jetson — Associate Professor, Dept of CSE, Kallam Haranadha reddy Institute of Tech

Dr. G. Satyanarayana Prasad — Professor & Dean, Dept of CSE, R.V.R& J.C college of Engineering

ABSTRACT

A variety of life cycle models for software development are generally available. However, it is generally difficult to compare and contrast the methods and very little literature is available to guide developers and managers in making choices. Moreover in order to make informed decisions developers require access to real data that compares the different models and the results associated with the adoption of each model. This paper describes an experiment in which fifteen software teams developed comparable software products using four different development approaches. Extensive measures were taken to assess the time, quality, size, and development efficiency of each product. The paper presents the experimental data collected and the conclusions related to the choice of method, its impact on the project and the quality of the results.

1.1 INTRODUCTION

Should the Waterfall model be dropped in favour of more modern approaches such as incremental development and eXtreme Programming (XP)? Many developers appear to show a preference for such modern approaches but there is not much published evidence to substantiate such choices. Software development and software engineering books discuss the alternatives but offer little in the way of direct comparison and explicit metrics that address the impacts on the product, project and people. Indeed metrics enable a deeper insight into the process by providing an empirical mechanism for objective evaluation.

The classic Waterfall model was refined in the early 1970s to cope with the larger and more demanding software projects characterized by a growing level of complexity. Many adaptations and adjustments have been made to the model leading to a variety of representations. Recent evidence suggests that it is still used extensively in many software development projects.

Chief among them are incremental methods, evolutionary methods , and more recently, XP and other agile methods. XP is often viewed as a lightweight methodology focused on the delivery of business value in small fully integrated releases that pass all the tests defined by the clients. Reduced focus on documentation and control allows development teams to produce more creative solutions more rapidly.

The aim of this work was to investigate the impacts of different approaches through the evaluation of the metrics generated through their use. Fifteen teams of student-developers working on similar applications utilized four development approaches ranging from a version of the waterfall model, via incremental and evolutionary development to Extreme Programming. The resulting data on the relative merits, attributes and features of the products, the time spent in each stage and the measures of requirements, design and programming outputs provide a start towards understanding the full impact of selecting a programming and management approach on the product, project, process and people and towards making better informed decisions about which one to apply under different circumstances.

1.2 BACKGROUND

The skill set focusing on the life cycle of software engineering projects is critical to both understanding and practicing sound development and management. Indeed life cycles are prominently featured within the Bodies of Knowledge of many different disciplines. The technical life cycle identifies the activities and events required to provide an effective technical solution in the most cost-efficient manner.

The benefits of using a life cycle approach is identified by:

- Visibility
- Modularity
- Task Identification
- Provision of Framework
- Control Of Project Finance
- Encouragement of Systematic evaluation
- Addressing the need for stakeholder review
- Providing a problem solving perspective
- Integrating activities
- Enable continuous monitoring,
- Managing uncertainty

Typical life cycle approaches to select from include sequential, incremental, evolutionary and agile approaches. Each is likely to be better suited to a particular scenario and environment and to result in certain impacts on the overall effort and the developed products.

Sequential Approaches. Sequential approaches (e.g. waterfall model, V-model) refer to the completion of the work within one monolithic cycle. Projects are sequenced into a set of steps that are completed serially and typically span from determination of user needs to validation that the given solution satisfies the user. Progress is carried out in linear fashion enabling the passing of control and information to the next phase when pre-defined milestones are reached and accomplished. This approach is highly structured, provides an idealized format for the contract and allows maximum control over the process. Sequential development is also referred to as serial engineering.

Incremental Approach

Incremental approaches emphasise phased development by offering a series of linked mini-projects working from a pre-defined requirements. Specification up front. Work on different parts and phases, is allowed to overlap throughout the use of multiple mini-cycles running in parallel. Each mini-cycle adds additional functionality and capability.

Evolutionary Approaches

Evolutionary approaches recognize the great degree of un-

certainty embedded in certain projects and allow developers and managers to execute partial versions of the project while learning and acquiring additional information and gradually evolving the conceptual design. Evolutionary projects are defined in a limited sense allowing a limited amount of work to take place before making subsequent major decisions. Projects can start with a macro estimate and general directions allowing for the fine details to be filled-in in evolutionary fashion.

Agile Approaches

Agile development is claimed to be a creative and responsive effort to address user's needs focused on the requirement to deliver relevant working business applications quicker and cheaper. The application is typically delivered in incremental fashion. The agile development approaches are typically concerned with maintaining user involvement through the application of design teams and special workshops.

1.3 THE EXPERIMENT

The experiment was designed to investigate the impact of a development approach on the resulting product and its attributes. At the outset it was hypothesized that the four life cycle models under investigation would result in similar project outcome. The experiment involved 55 student-developers working in fifteen teams developing comparable products from the same domain. The objective of the experiments was to investigate the differences in terms of development effectiveness and quality given the different approaches. Of the 15 projects all groups were asked to use JAVA and J2EE for transaction processing. Most used Sun public domain server tools for the deployment of J2EE and the Cloudscape database. A small number of the groups used JBoss server, MySQL database system and Borland development tools. Regardless of the platform selected, the groups were instructed to experiment their way through the technology with limited formal help. The result of the effort was a working product that can be viewed as working usability or technology prototype.

1.4. RESULTS

This section describes and analyses the direct measurements from the different teams as well as some of the derived metrics related to the four development approaches. The numerical results are presented in Tables 1 – 4 in the form of group averages. The underlying values, i.e. the individual values for the groups, are displayed.

As part of the statistical analysis to establish the significance of the results, all of the project metrics were analysed using the R Statistical Package. In order to reason about the significance of the differences in the group averages a null hypothesis is postulated:

H0 -The averages for a particular attribute for the four models are alike when compared pair wise.

In this work the H0 is rejected "with 90% confidence" if the "p statist ic" assumes a value less than 0.1 (p < 0.1) in pair wise comparisons using t tests with pooled standard deviation. In tables 1 – 7 (as well as in Tables 1 - 4) statistically significant differences in pair wise comparisons for averages are shown in bold. It turns out that all of the significant differences relate to the XP model. It is to be note that the sample sizes n (i.e. the groups representing each method) are small, n = 3 or n = 4, so that the differences in the averages must be quite marked in order to be significant. Discussions in the paper are focussed on differences that are statistically significant.

One of the Incremental groups spent 8.4 Project Months (PM) (1275 hours) on the project. This group is taken to be a statistical outlier as their result is out of line with other groups and the overall time availability and is therefore dropped from the tables below. (Rationale: The value of 8.4 PM is well above the likely maximum of 6.6 PM as computed by the rule of thumb: median + 1.5 Interquartile_Range.) The overall average of the effort spent for the remaining groups is then 3.9 PM.

Table 1 shows the average effort data by development approach. Time spent is measured in hours and is divided into the seven general activities of requirements specification, design, coding, testing, review, repair and other giving an overall total for each project. The total is converted into Project Months (PM) at 152 hours per PM. The groups are ordered by the development model used – V-model (VM), Evolu tionary Model (EM), Incremental Model (IM), and Extreme Programming (XP). The overall average (OAve) is also presented.

The only significant difference in the group averages (in bold) are found in the requirements specification activity where the XP groups spent considerably less time (16.2) as compared to that of the VM groups (73.9) and the EM groups (67.8) (i.e. the H0 hypothesis of equal averages is rejected with 90% certainty.) Italics are used to signify the base of the comparison. In other words: XP < VM and XP < EM .

Table 1 Average group effort in hours by activity

Group	*Req.Spec.*	Design	Code	*Integr. & Test*	Review	Repair	Other	*Total hours*	*Total PM*
VM	**73.9**	68.6	206.1	35.6	56.9	60.4	246.5	748.0	4.92
EM	**67.8**	58.0	169.1	57.8	23.0	48.0	125.6	549.3	3.61
IM	43.8	51.2	185.7	40.7	37.7	53.8	121.6	534.5	3.52
XP	***16.2***	26.2	205.6	82.7	46.9	92.7	122.4	592.5	3.90
OAve	53.3	52.8	191.1	53.1	41.0	62.4	158.6	612.1	4.03

Figure 1 shows boxplots for the total hours of the 14 teams. While the boxplots for the methods appear to vary considerably, the differences in the overall averages are not significant. Clearly there is quite a large difference in the averages for individual group effort values. The longest time spent by a group, 919 hours, belongs to a V-model group whilst the shortest period, 373 hours, was exerted by an XP group (See Table 1A).

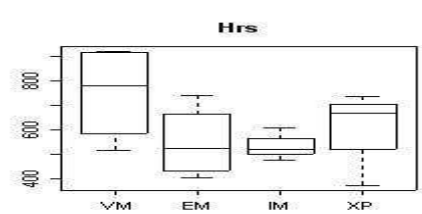


Figure 1 Distribution of total hours spent

The amount of work performed by each student varied considerably as is often the case in student projects. Furthermore few of the groups consisted of only three members while most had four. To counteract these differences it is seen appropriate to work with the percentile distribution of effort in order to reason about the effort patterns for the different methods.

Table 2 shows the group effort by activity as percentages. As noted for Table 1, the only significant difference in the

Volume : 5 | Issue : 7 | July 2016 • ISSN No 2277 - 8179 | IF : 3.508 | IC Value : 69.48

averages is for the requirements specification effort.The XP groups' values (3%) were significantly lower than that exerted by the VM groups (10%) and the EM groups (12%) as was the case in Table 1.

Table 2 Average group effort by activity as percentages

Group	ReqSpec %	Design %	Code %	Integr. & Test %	Review %	Repair %	Other %	Total %
VM	10	10	27	5	7	8	33	100
EM	12	10	29	11	5	9	23	99
IM	8	9	35	7	7	10	23	99
XP	3	4	34	14	8	17	21	101
OAve	8.9	8.9	30.8	8.9	6.5	10.0	25.7	100

Other activities than requirements specification are statistically alike even if they show considerable fluctuation. It is noteworthy that the coding activity consumed 30.8% of the overall effort with the group averages closely aligned.

B. Requirements and high level design output

The output of the requirements specification and high level design are shown in Table 3 as assessed in terms of the number of pages, Use Cases, screens, database diagrams, and DB tables. Note that Use Cases and XP "stories" were taken to be equivalent.

The XP groups produced an average of only 5.7 pages of requirements specification

Table 3 Requirements and high level design outcome

Group	Pages	Use cases	Screens	DB diagrams	DB Tables
VM	22.0	7.5	6.8	1.0	3.3
EM	19.0	14.0	11.0	1.3	6.8
IM	22.7	14.7	5.0	1.3	4.3
XP	5.7	8.3	17.7	0.7	5.0
OAve	17.8	11.1	9.9	1.1	4.9

which is significantly lower than that of any of the VM, EM or IM groups (at 22.0, 19.0 and 22.7 respectively.) The XP groups did produce, on the other hand, a significantly higher number of user interaction screens (17.7) than the VM and IM groups (6.8 and 5.0).

C. Code size

The code size (Tables 5 and 5A) is measured in terms of Java classes and lines of code. Lines of code include lines produced in JAVA as well as JSP, XML and any other scripting code.

Table 4 Number of classes and lines of code

Group	Java classes	Java	jsp	XML	Other	Total LOC
VM	8.5	1032	1161	13	27	2233
EM	26.8	1477	1903	0	37	3417
IM	20.7	1803	922	36	14	2776
XP	27.7	4836	1662	987	254	7740
OAve	20.4	2140	1429	223	76	3867

In terms of Java lines of code, the XP model teams delivered significantly more code than the other models with an average of 4836 compared with 1803 LOC from the Incremental teams, 1477 from the Evolutionary and 1032 from the V-Model. The group averages for all other code are statistically alike.

D. Defects

Each team was meant to collect defect data and classify it into 7 types of defects classified according to the Orthogonal Defect Classification (ODC) scheme . This data turned out to be heterogeneous and inconsistent and is therefore not reported.

1.5. CONCLUSION

The results of this experiment provide some useful quantitative information on the relative merits of different development methods and on the impact they can have on different activities. The results can be sensitive to the performance of individuals within groups and to the synergy created within any group, as there were typically only four groups of each life cycle approach.

Despite the fact that all projects addressed comparable problems, V-model projects took somewhat longer than the other projects. The requirements stage in the V-model consumed more time than the requirements activities in the other models. Similar trends were observed in design activities. In terms of the significance of the results, XP only consumed a minor proportion of the effort during the requirements activities compared to the other models.

In terms of all projects, the activity of requirements was responsible for roughly 9% of the time spent. The activity of design also consumed approximately 9% of the overall time. In XP however, requirements resulted in 3% and design in 4%. Overall, programming used up 31% of the time, integration and testing 9%, reviewing 7%, repair 10% and other activities 26%. It is important to note that activities not directly considered as part of the life cycle ('other') still accounted for over a quarter of the time and must therefore be planned for in the early stages of the project. This is roughly in line with industrial rules of thumb for additional (and non-technical) activities. Note however that the time spent on 'other' activities by V-model teams was on average double the time spent by the other teams reflecting the additional effort and adjustments resulting from deferring the interaction with technology to the later stages.

REFERENCES

1. Bennington, H.D., Production of Large Computer Programs. Annals of the History of Computing, 1956. 5 Oct. 1983(4): p. 350-361.
2. Royce, W.W. Managing the Development of Large Software Systems: Concepts and Techniques. in Proceedings, IEEE WESCON, August 1970. 1970.
3. Farr, L.A., Description of the Computer Program Implementation Process. 1963, SDC Technical Report.
4. Laplante, P.A. and C.J. Neill, The Demise of the Waterfall Model is Imminent and other Urban Myths. ACM Queue, 2004. 1(10): p. 10-15.
5. Pressman, R.S. and D. Ince, Software Engineering: A Practitioner's Approach. 5 ed. 2000, Maidenhead: McGraw-Hill.
6. Chatters, B.W. Software Engineering: What do we know? in FEAST 2000, July 2000. 2000. Imperial College, London.
7. Cusumano, M.A., et al., A Global Survey of Software Development Practices. 2003, MIT: Cambridge, Ma. p. 1-17.
8. Mills, H.D., Incremental Software Development. IBM Systems Journal, 1980. 19(4): p. 415-420.
9. Beck, K., Extreme Programming Explained: Embrace Change. 2000, Boston, MA: Addison-Wesley.
10. Laraman, C. and V.R. Basili, Iterative and Incremental Development: A Brief History. IEEE Computer, 2003. 36(6): p. 47-56.
11. Layman, L. Empirical Investigation of the impact of Extreme Programming Practices on Software Projects. in OOPSLA '04. 2004. Vancouver, British Columbia: ACM Press.