ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 4, Special Issue 11, September 2015

National Conference on Emerging Trends in Computing, Communication & Control Engineering [NCET 2K15]

On 4th September, 2015

Organized by

Department of CSE, ECE & EEE, Magna College of Engineering, Chennai-600055, India.

Web Development Program Equivalence by an Explorative Try out Searching for Emering Program Properties

T.Durga[1], V.Rajkumar[2]

Assistant Professor, Dept of CSE, Sri Venkatesa Perumal College of Engineering, Andhra Pradesh, India [1]

Assistant Professor, Dept of IT, Krishnasamy College of Engineering & Technology, Cuddalore, Tamil Nadu, India [2]

ABSTRACT: In the context of Web-based application development, a technological Program is a programming language plus the set of reusable technological pieces used in conjunction with that language, such as components, frameworks, libraries, tools, and auxiliary languages. A technological ecosystem consisting of a programming language (and possibly alternatives), one or more Web development frameworks, and a large set of reusable libraries and components and development culture consisting of styles, priorities, process preferences, etc. Such comparisons can sometimes be helpful when making platform decisions, but cannot provide objective information about the platforms' emergent characteristics. Many informal comparisons of Web development platforms or frameworks exist. There are results that many people would have called "obvious" or "well known," say that Perl solutions tend to be more compact than Java solutions. Second, there are results that contradict conventional wisdom, say, that our PHP solutions appear in some (but not all) respects to be actually at least as secure as the others. Finally, one result makes a statement we have not seen discussed previously: Along several dimensions, the amount of within-platform variation between the teams tends to be smaller for PHP than for the other platforms. Most of them are not based on actually trying out the frameworks, however, but rather compare features and programming styles theoretically. Regarding the platform differences found, one could say that we have learned a lot, but the question is certainly rather far from settled. We conclude that the overall approach of Platforms is feasible and appears worthwhile. However, with respect to actually evaluating the individual criteria, a lot of detail remains to be worked out—several of the evaluations turned out to be exceedingly difficult.

KEYWORDS: Emergent properties, usability, functionality, reliability, security, product size, design structure.

I. INTRODUCTION

1.1 Background: Web Development Platforms

HUGE numbers of projects of all sizes for building Web based applications of all sorts are started each year. However, when faced with selecting the technology ("platform") to use, more than one platform may appear sensible for a given project. In the context of Web-based application development, a technological platform is a programming language plus the set of reusable technological pieces used in conjunction with that language, such as components, frameworks, libraries, tools, and auxiliary languages. Web development platforms share a number of auxiliary languages, in particular (X) HTML, CSS, and JavaScript. However, the technology is not the only thing that is criminates platforms because most of them bring along their own distinct development culture as well. For the purpose of this work, we define platforms (meaning Web development platforms) as follows: platform is the combination of A technological ecosystem consisting of a programming language (and possibly alternatives), one or more Web development frameworks, and a large set of reusable libraries and components and A development culture consisting of styles, priorities, process preferences, etc.

1.2 Research Question

The technology is not the only thing that discriminates platforms because most of them bring along their own distinct development culture as well. Platform characteristics exist that consistently differ from one platform to another or do

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 4, Special Issue 11, September 2015

National Conference on Emerging Trends in Computing, Communication & Control Engineering [NCET 2K15]

On 4th September, 2015

Organized by

Department of CSE, ECE & EEE, Magna College of Engineering, Chennai-600055, India.

the differences between development teams mask such differences. Diagnosing consistent platform differences (if they exist) will be dramatically easier if we have a high amount of control of the setting, so an experimental research approach is much preferred to a merely observational one such as, for instance, case studies. To understand the relative pros and cons of different Web development platforms, characterize all consistent differences in the development processes used with these platforms and the products they bring forth with small teams of professional developers for medium-small, stand-alone Web applications, from the perspective of developers, modifiers, customers, users, and researchers using a replicated project study. Platforms is thus a nonrandomized controlled experiment quasi-experiment, with one categorical independent variable (the platform) with three levels Java, Perl, and PHP and a large number of dependent variables as mentioned. It controls most human-related variables by averaging using teams of three and using three teams per platform), but accepts selection effects as natural "appropriate self selection

1.3 Platforms Overview

In the standard language of [2], Platforms is characterized like this: To understand the relative pros and cons of different Web development platforms, characterize all consistent differences in the development processes used with these platforms and the products they bring forth with small teams of professional developers for medium-small, stand-alone Web applications, from the perspective of developers, modifiers, customers, users, and researchers using a replicated project study. More concretely, Platforms is an event in which several professional Web development teams meet in one place at one time to create an implementation for the same requirements within a fixed time frame, each team using the platform they are most highly skilled with.

1.4 Evaluation Overview

In light of the research question, Platforms attempts to evaluate as many characteristics as possible: We will assess platform productivity differences by looking at the completeness of the solutions submitted and the participants' subjective platform experience. We will assess platform differences with respect to the development process by looking at participant activity time series data and the source code archive version history. We will assess external quality factors by looking at the solutions' ease-of-use, robustness and security, correctness, and scalability). We will assess the maintainability related internal quality factors productivity (by looking at product size,) modularity, and modifiability.

II. PLAT_FORMS SETUP

2.1 Platforms as an Experimental Design Platforms is an attempt to perform a controlled experiment: Vary one factor (the "independent variable," here, the platform), keep everything else constant ("control all other variables"), and see how the various resulting project Characteristics ("dependent variables," such as size, quality, efficiency, etc.) change. This approach to observation is a fundamental principle of the scientific method. When humans play a major role in an experiment, keeping "everything else constant" is usually possible only via repeated measurement (and averaging over several humans) in order to smooth out individual characteristics. The repetition has to be combined with random assignment of the people to the various experiment conditions ("randomization") in order to avoid self-selection effects, say, if all of the most capable people favored platform X [4]. In the Platforms case, however, the last point is different. The following decisions were made in order to make the Platforms setup as credible as possible: use a task of somewhat realistic size and nature,. Use teams of three as subjects rather than individuals, Use a setting that is more similar to real project conditions than to laboratory conditions,. Refrain from randomization. The last point requires elaboration. For a platform comparison such as ours, randomization would be problematic because it would make the setting unrealistic: A real project owner would be unwise to let a team work with a platform that is chosen independently of the team. Rather, one would either choose a team or let it work with the platform for which it has the best skills or choose a platform, and then find an appropriate team that is skilled with it. We have no interest in learning about what happens when teams work with random platforms.

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 4, Special Issue 11, September 2015

National Conference on Emerging Trends in Computing, Communication & Control Engineering [NCET 2K15]

On 4th September, 2015

Organized by

Department of CSE, ECE & EEE, Magna College of Engineering, Chennai-600055, India.

2.2 Requirements Process

The teams could ask for clarifications at any time. Apart from a few minor imperfections of the requirements document, this option was hardly used though. In particular, there were hardly any questions regarding the relative priorities of different nonfunctional properties.

2.3 Threats to Validity

Many aspects of a real project are not reflected in the task, such as requirements management, documentation, acceptance testing, installation, administrator training, etc. Adding these aspects would bring the task size to a range of about two or three person months, which is still small for a professional software project (yet big enough to bse taken seriously). It would also allow additional candidate platform differences to be analyzed, which are missing in this study. The lack of these aspects may have influenced some of our results somewhat. In particular, our study does not involve requirements changes during project execution, which plausibly might have influenced correctness and maintainability.

III. RELATED WORKS

3.1 Platform Comparisons

Many informal comparisons of Web development platforms or frameworks exist. Most of them are not based on actually trying out the frameworks, however, but rather compare features and programming styles theoretically. Such comparisons can sometimes be helpful when making platform decisions, but cannot provide objective information about the platforms' emergent characteristics. There are also a number of comparisons that involve actual programming, but they usually differ sharply from Platforms with respect to one or more of the following important aspects:

- Many of them involve much less controlled conditions for the production of the solutions. Authors can put in an arbitrary amount of work during the course of several weeks.
- Many of them focus on only a single evaluation criterion, such as performance or the length of the program code.
- Some are prepared by a single author only, which raises the question whether we can assume that a similar level of platform-specific skills was applied for each platform.

IV. EVALUATION RATIONALES

This section provides a quick overview of the various facets of our evaluation. For each aspect studied, we describe what it is, why it is of interest, the evaluation approach, and the expectations of what we might find and why. The results for each aspect will be given in a separate section (or sometimes several) , each also containing more details on the evaluation method. We will evaluate platform productivity by looking at the completeness of the solutions in terms of how many of the fine-grained requirements have been realized.Productivity is a fundamental attribute for efficient software engineering. We might expect, for instance, that Java Is less productive than the others because of its generally more heavyweight nature.3 We attempt to find differences in the development processes used (without judging them as good or bad) by Analyzing observations of roughly what each team member was doing when (recorded in 15 minute intervals, Section 5.2) and by analyzing the version repositories for salient differences of any kind (Section 5.3). Process differences might indicate for which kinds of projects or customer temperaments a platform may be most suitable. We might expect, for instance, to find evidence that the processes of the Java teams tend to be more coarse-grained or waterfallish and those of the other teams more incremental and iterative. We informally evaluate ease-of-use (Section 5.4, an aspect of usability) by subjectively judging each of the 108 GUI requirements. This might indicate platforms whose culture provides better user focus or whose technology Provides better support for ease-of-use. We have no specific expectations in this regard. We evaluate the applications' robustness, error handling, and (superficially) security by various user-level black box tests. This might tell platforms with a less well-developed quality culture or platforms whose technology imposes a higher programming burden for achieving appropriate data checking and filtering. We expect Java to be the most problem-aware platform in these respects and PHP to be the worst.

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 4, Special Issue 11, September 2015

National Conference on Emerging Trends in Computing, Communication & Control Engineering [NCET 2K15]

On 4th September, 2015

Organized by

Department of CSE, ECE & EEE, Magna College of Engineering, Chennai-600055, India.

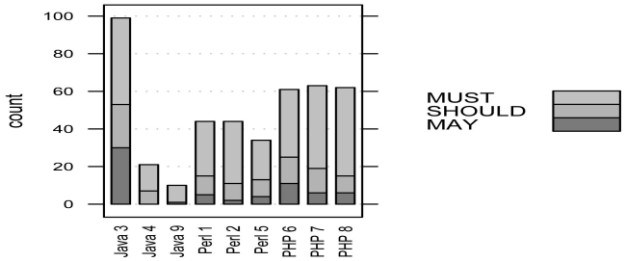

Fig. 1. Completeness of solutions in terms of the number of UI-related requirements implemented that are working correctly (grades 2, 3, and 4).

and works correctly; (2) is implemented, but works less well (in particular, with worse ease-of-use) than expected;(1) is only partially implemented or does not work properly; and (0) is not implemented. The two judges (who were graduate computer science students and performed the judgment mostly based on their common sense as experienced computer users) then discussed and resolved any discrepancies between their judgments to arrive at the unified judgment used in the evaluation below. Across the nine solutions thus judged, the median agreement of the judges before the discuss-and-resolve procedure was 80 percent. We count a requirement as implemented if its grade is higher than 1. Later on, we will use the discrimination of grades 0 versus 1 for judging correctness and the discrimination of grades 2 versus 3 versus 4 for judging ease-of-use. We used a simplified procedure for the Web service requirements 109 through 127 because those can be judged more canonically: Only one judge performed the assessment. Mark 4 was not used; mark 2 was only used to designate slight defects. We use these data for evaluation in two different ways: once by merely counting the implemented requirements and once by weighting them, each according to the effort presumably required for implementing this requirement. Weights ranged from 0 points for redundant compound requirements (whose content is present again in separate detailed requirements) and 1 point for trivial requirements (such as simple individual input fields) up to 5 points for difficult functionality

V. RESULTS

5.1 Results from Activity Indicators

We investigated the activity indicator data in very many different ways: using numerical summaries or graphical plots; using raw status codes or status kinds or commstatus; on the level of individual, team, or platform; on a granularity of 15 minutes, a few hours, or the whole contest. We found a number of peculiarities on the team level. For instance (to mention a few of the extremes), Web browser usage was about 10 times as frequent for team1 Perl as for team9 Java; communication among people was more than twice as frequent for team8 PHP than for team7 PHP, let alone team3 Java; team1 paused only half as much as the average team—you can see the latter facts in Fig. 2.But, when we aggregated the teams into platforms, no consistent differences remained.

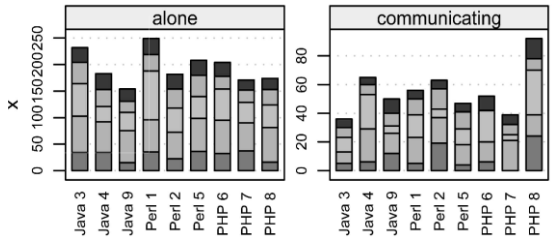

Fig. 2. Commstatus frequencies by team for five major phases of the contest, namely, start (9:00-13:00 hours), day1 (13:00-20:00 hours), night (20:00-6:00 hours), morning2 (6:00-12:00 hours), end (12:00-15:00 hours).

5.2 Results Regarding Questions to the Customer

Team members approached the customer between once (team9 Java) and 13 times (team6 PHP) per team. Fig. 3 shows the distribution of question types over the platforms. The overall differences might be expected (at least in comparison

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 4, Special Issue 11, September 2015

National Conference on Emerging Trends in Computing, Communication & Control Engineering [NCET 2K15]

On 4th September, 2015

Organized by

Department of CSE, ECE & EEE, Magna College of Engineering, Chennai-600055, India.

to Java) by people who believe that using a dynamic language tends to imply using more agile development attitudes [8], and hence, stronger customer-orientation. However, the given differences are not large enough to be considered clear-cut evidence in this respect. It remains unclear whether there are platform differences or not.

5.3 Threats to Validity

The fact that we found no platform differences does not mean there aren't any. The set of status codes used was dictated by feasibility constraints and may have hidden the actual differences. However, a set of codes that could provide deeper insights into the process aspects mentioned Above would have required such an immensely higher

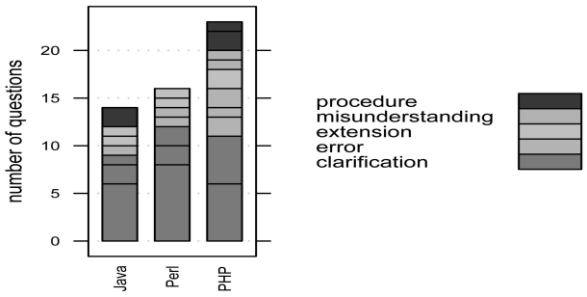

Fig. 3. How often team members from each platform inquired of the customer, separated by the topic or reason of the inquiry. The number of inquiries is not large enough to find an interesting structure in a bar plot by teams or in a density plot per team running over the course of the 30 hours. The subjecting lines indicate the individual teams.

5.5 Robustness, Error Handling, and Security: Differences Found

5.5.1 Data Gathering Method

Fully assessing robustness and security is an extremely laborious process. Furthermore, if white-box techniques such as code reviews are employed, it becomes very difficult to ensure that, despite the huge differences in programming language and application structure, each Weakness has the same chance of being uncovered. To ensure both feasibility and fairness, we thus resorted to running simple user-level black-box tests only, as follows:

. < = . . . >: handling of HTML-tags embedded in user

Input (to assess the danger of cross-site scripting

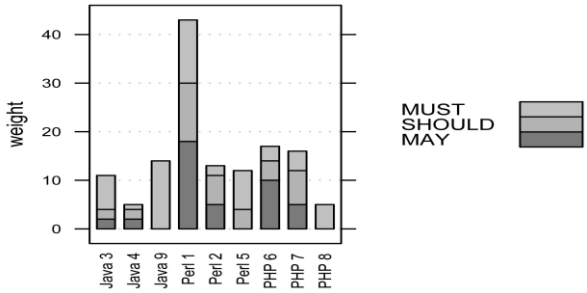

Fig. 4. Number of UI requirements whose implementation received a grade of 1 ("incorrect") during completeness testing, meaning the implementation malfunctions massively (i.e., in a quarter of all cases)

- long: handling of very long user inputs
- int'l: handling of Chinese ideograms embedded in user inputs
- email: e-mail address validity checking
- SQL: handling of quotes embedded in user inputs (to assess simple attempts at SQL injection)
- cookie: operation with cookies turned off

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 4, Special Issue 11, September 2015

National Conference on Emerging Trends in Computing, Communication & Control Engineering [NCET 2K15]

On 4th September, 2015

Organized by

Department of CSE, ECE & EEE, Magna College of Engineering, Chennai-600055, India.

5.6 Performance and Scalability: Unassessable

Scalability refers to the degree to which a solution is capable of providing rapid response times despite large numbers of concurrent users and requests. We had intended to evaluate scalability by means of load testing. Since each of the nine solutions had a completely different HTML user interface, designing and implementing a fair and meaningful load test for each of them was beyond our resources. We had therefore planned to perform load testing via the Web service interface, which should have been identical for each solution. Unfortunately, only two of the solutions implemented the Web service interface to a sufficient degree

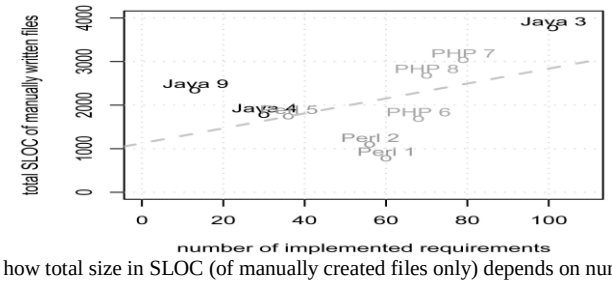

Fig. 5. Linear regression showing how total size in SLOC (of manually created files only) depends on number of requirements implemented.

- The Perl solutions have fewer lines-of-code in manually created files than the PHP solutions (the difference is 487 to 2,035 SLOC with 80 percent confidence) and tend to have fewer than the Java solutions (the difference is 332 to 2,511 SLOC).
- When normalizing these numbers by dividing by the number of requirements implemented by each team, there remains a tendency that Perl solutions and PHP solutions require fewer manual lines of code (averages: 27 and 34) per requirement than Java (average: 92), but the variance is too high to detect a consistent trend.
- A linear regression for these data (size depending on functionality) indicates that Java solutions tend to be less compact and Perl solutions more compact than the average (see Fig. 5). Summing up, we find that the Perl solutions were the smallest ones, both in absolute terms and relative to their functionality, and that they were also the most template driven ones.

5.6.1 Threats to Validity

The degree of modification in the generated-but-modified, and in particular, the reused-but-modified files varies a lot. It might have been more appropriate if some of these had in fact been counted as manually created files. We do not expect the difference to be large enough to change the results, though.

VI. SUMMARY, AND CONCLUSION

6.1 Platform-Related Differences Observed

6.1.1 Java-Centric Differences

- Java was the only platform for which all three solutions handle HTML tags in text field input so as to avoid client-side XSS security issues.
- The amount of functionality implemented by the Java teams is much less uniform than for the others. Also, the nature of the changes required for adding a user profile data field into the application was more dissimilar among the Java solutions than among the solutions of the other platforms.

6.1.2 Perl-Centric Differences

- The Perl solutions are smaller than the Java and the PHP solutions.
- The number of changes required for adding a user profile data field into the application was smallest for Perl.

6.1.3 PHP-Centric Differences

- The amount of functionality implemented by the PHP teams is larger than that of the Perl teams and more uniform than it is for both other platforms.

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 4, Special Issue 11, September 2015

National Conference on Emerging Trends in Computing, Communication & Control Engineering [NCET 2K15]

On 4th September, 2015

Organized by

Department of CSE, ECE & EEE, Magna College of Engineering, Chennai-600055, India.

- A similar statement holds for several other criteria as well: The PHP teams are more similar to one another in those respects than are the teams from the other platforms with respect to the fraction of requirements implementations that have low quality from the user's point of view, the size and composition of the source distribution, or the number of lines of code required on average for implementing one of the requirements.

6.1.4 Summary

The picture painted by the empirical differences above is diverse. There are various ways to summarize these results: One may focus on those results that confirm what some people may consider to be common expectations, such as "Perl solutions are lightweight," "Java solutions are heavyweight, "and "A good commercial framework can provide higher productivity than today's open source frameworks do" (team3 Java was the only one to rely on a commercial framework). One may focus on those results that contradict such expectations such as "The PHP solutions are at least as secure as the others." One may take a risk-minimization view and point out that, over a range of different criteria, the Java solutions exhibited the largest variation (meaning high risk), while the PHP solutions had the smallest (hence, lower risk).Finally, one may point out that, overall, the results suggest that the manner in which a platform is used is more important than the platform technology as such. For instance, in the maintainability scenarios, the Java solutions were much more complex than the Perl solutions, but this was an arbitrary decision by the programmers; they could as well have used the same approach as the Perl teams. The main conclusion in this case would be "People matter more than platform technologies."

REFERENCES

[1] B.C.D. Anda, D.I.K. Sjøberg, and A. Mockus, "Variability and Reproducibility in Software Engineering: A Study of Four Companies that Developed the Same System," IEEE Trans.Software Eng., vol. 35, no. 3, pp. 407-429, May/June 2009.
[2] V.R. Basili, R.W. Selby, and D.H. Hutchens, "Experimentation in Software Engineering," IEEE Trans. Software Eng., vol. 12, no. 7,pp. 733-743, July 1986.
[3] V.R. Basili, F. Shull, and F. Lanubile, "Building Knowledge through Families of Experiments," IEEE Trans. Software Eng.,vol. 25, no. 4, pp. 456-473, July/Aug. 1999.
[4] L. Christensen, Experimental Methodology, 10th ed. Allyn and Bacon, 2006.
[5] A. Cockburn, Writing Effective Use Cases. Addison-Wesley, 2001.
[6] B. Curtis, "By the Way, Did Anyone Study Any Real Programmers?"Proc. First Workshop Empirical Studies of Programmers, pp. 256-262, 1986.
[7]A. Green and B. Askins, "A Rails/Django Comparison," http://docs.google.com/View?docid=dcn8282p_1hg4sr9, Sept. 2006.
[8] J. Highsmith and A. Cockburn, "Agile Software Development:The Business of Innovation," IEEE Software, vol. 18, no. 5, pp. 120-122, Sept./Oct. 2001.
[9] V.B. Kampenes, T. Dyba°, J.E. Hannay, and D.I.K. Sjøberg, "A Systematic Review of Quasi-Experiments in Software Engineering,"Information and Software Technology, vol. 51, no. 1, pp. 71-82,Jan. 2009.
[10] S. Kelly, "Better Web App Development," Quicktime Video, http://oodt.jpl.nasa.gov/better-web-app.mov, Feb. 2006.
[11]M. Kunze and H. Schulz, "Gute Nachbarschaft: C't La¨dt zum Datenbank-Contest Ein," C't, 20/2005:156, 2005. see also http://www.heise.de/ct/05/20/156/, english translation on http://firebird.sourceforge.net/connect/ct-dbContest.html, overview on http://www.heise.de/ct/dbcontest/, results in issue 13, 2006.
[12] D. Mu¨ ller-Solger, "Wettbewerb der Portale," IT Management,vol. 9, pp. 47-48, 2006.
[13] Usability Inspection Methods, J. Nielsen and R.L. Mack, eds. JohnWiley and Sons, 1994.[14] J.M. Perpich, D.E. Perry, A.A. Porter, L.G. Votta, and M.W. Wade,"Anywhere, Anytime Code Inspections: Using the Web to Remove Inspection Bottlenecks in Large-Scale Software Development,"Proc. 19th Int'l Conf. Software Eng., pp. 14-21, 1997.