



UNIVERSITI PUTRA MALAYSIA

**GRAPHICAL USER INTERFACE LAYOUT LANGUAGE USING
COMBINATORS**

KHAIRUL AZHAR KASMIRAN

FSKTM 2006 3

**GRAPHICAL USER INTERFACE LAYOUT LANGUAGE USING
COMBINATORS**

By

KHAIRUL AZHAR KASMIRAN

**Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia,
in Fulfilment of the Requirements for the Degree of Master of Science**

March 2006



Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirement for the degree of Master of Science

GRAPHICAL USER INTERFACE LAYOUT LANGUAGE USING COMBINATORS

By

KHAIRUL AZHAR KASMIRAN

March 2006

Chairman: Associate Professor Abdul Azim Abd. Ghani, PhD

Faculty: Computer Science and Information Technology

While Java is a popular general-purpose programming language, there are certain areas where the syntax of Java is lacking for the task at hand. One of them is in the area of layout handling, i.e., the task of placing controls in a Graphical User Interface (GUI) with regard to their relative position and size. This is because the syntax of Java is targeted towards imperative programming, where code is written in the form of a list of instructions. A list of instructions does not adequately mirror the hierarchical structure of a layout.

To overcome that weakness, this thesis describes and evaluates a new domain-specific programming language designed specifically for layout handling, named Swing GUI Layout Language (SGLL). One of the primary features of SGLL is the use of combinators, a concept used in functional languages. We propose that combinators are a more intuitive concept compared to the approach taken by Java, which involves adding controls to a layout manager. Furthermore, we suggest that elimination of clutter and better support for the abstractions in layout handling can provide an increase in programmer productivity and understandability of the source

code. In this thesis, we focus on the GridLayout manager class, since it is rather easy to understand and provides a good starting point.

To validate our approach, we evaluated Java and SGLL in both productivity and understandability. We found out that SGLL does provide a significant improvement in productivity and understandability for the task of layout handling.

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Master Sains

**BAHASA PENGATURCARAAN SUSUN ATUR MENGGUNAKAN
PENGHUBUNG UNTUK ANTARAMUKA GRAFIK**

Oleh

KHAIRUL AZHAR KASMIRAN

Mac 2006

Pengerusi: Profesor Madya Abdul Azim Abd. Ghani, PhD

Fakulti: Sains Komputer dan Teknologi Maklumat

Walaupun Java merupakan bahasa pengaturcaraan serbaguna yang popular, masih ada beberapa bidang di mana sintaks Java kurang sesuai untuk tugas yang diberi. Salah satu daripadanya adalah dalam bidang pengendalian susun atur, iaitu tugas meletakkan alat kawalan dalam satu antara muka pengguna grafik (Graphical User Interface) dengan merujuk kepada kedudukan sesama mereka dan saiz mereka. Ini adalah kerana sintaks Java lebih tertumpu kepada pengaturcaraan imperatif, di mana kod ditulis dalam bentuk satu senarai arahan. Satu senarai arahan kurang mencerminkan struktur hierarki sesebuah susun atur.

Untuk membetulkan kelemahan tersebut, tesis ini menerang dan menilai satu bahasa pengaturcaraan domain-khusus baharu yang direka bentuk khas untuk pengendalian susun atur, dinamakan SGLL. Salah satu ciri utama SGLL adalah penggunaan penghubung, yang merupakan satu konsep yang digunakan dalam bahasa pengaturcaraan fungsian. Kami merasakan bahawa penghubung merupakan satu konsep yang lebih senang difahami berbanding pendekatan yang digunakan oleh Java, yang melibatkan penambahan alat kawalan kepada sebuah pengurus susun atur. Tambahan pula, kami percaya bahawa penyingkiran penyelerakan dan sokongan

yang lebih baik untuk pengabstrakan dalam pengendalian susun atur dapat meningkatkan produktiviti pengaturcara dan pemahaman kod sumber. Dalam tesis ini, kami memberikan tumpuan kepada kelas pengurus GridLayout, memandangkan kelas tersebut senang untuk difahami dan merupakan satu titik permulaan yang baik.

Untuk mengesahsahihkan pendekatan kami, kami telah menilai Java dan SGLL dalam kedua-dua produktiviti dan pemahaman. Kami mendapati bahawa hasil kami menunjukkan bahawa SGLL dapat meningkatkan produktiviti dan pemahaman untuk tugas pengendalian susun atur.

**GRAPHICAL USER INTERFACE LAYOUT LANGUAGE USING
COMBINATORS**

By

KHAIRUL AZHAR KASMIRAN

**Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia,
in Fulfilment of the Requirements for the Degree of Master of Science**

March 2006



Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment
of the requirement for the degree of Master of Science

**GRAPHICAL USER INTERFACE LAYOUT LANGUAGE USING
COMBINATORS**

By

KHAIRUL AZHAR KASMIRAN

March 2006

Chairman: Associate Professor Abdul Azim Abd. Ghani, PhD

Faculty: Computer Science and Information Technology

While Java is a popular general-purpose programming language, there are certain areas where the syntax of Java is lacking for the task at hand. One of them is in the area of layout handling, i.e., the task of placing controls in a Graphical User Interface (GUI) with regard to their relative position and size. This is because the syntax of Java is targeted towards imperative programming, where code is written in the form of a list of instructions. A list of instructions does not adequately mirror the hierarchical structure of a layout.

To overcome that weakness, this thesis describes and evaluates a new domain-specific programming language designed specifically for layout handling, named Swing GUI Layout Language (SGLL). One of the primary features of SGLL is the use of combinators, a concept used in functional languages. We propose that combinators are a more intuitive concept compared to the approach taken by Java, which involves adding controls to a layout manager. Furthermore, we suggest that elimination of clutter and better support for the abstractions in layout handling can provide an increase in programmer productivity and understandability of the source

code. In this thesis, we focus on the GridLayout manager class, since it is rather easy to understand and provides a good starting point.

To validate our approach, we evaluated Java and SGLL in both productivity and understandability. We found out that SGLL does provide a significant improvement in productivity and understandability for the task of layout handling.

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Master Sains

**BAHASA PENGATURCARAAN SUSUN ATUR MENGGUNAKAN
PENGHUBUNG UNTUK ANTARAMUKA GRAFIK**

Oleh

KHAIRUL AZHAR KASMIRAN

Mac 2006

Pengerusi: Profesor Madya Abdul Azim Abd. Ghani, PhD

Fakulti: Sains Komputer dan Teknologi Maklumat

Walaupun Java merupakan bahasa pengaturcaraan serbaguna yang popular, masih ada beberapa bidang di mana sintaks Java kurang sesuai untuk tugas yang diberi. Salah satu daripadanya adalah dalam bidang pengendalian susun atur, iaitu tugas meletakkan alat kawalan dalam satu antara muka pengguna grafik (Graphical User Interface) dengan merujuk kepada kedudukan sesama mereka dan saiz mereka. Ini adalah kerana sintaks Java lebih tertumpu kepada pengaturcaraan imperatif, di mana kod ditulis dalam bentuk satu senarai arahan. Satu senarai arahan kurang mencerminkan struktur hierarki sesebuah susun atur.

Untuk membetulkan kelemahan tersebut, tesis ini menerang dan menilai satu bahasa pengaturcaraan domain-khusus baharu yang direka bentuk khas untuk pengendalian susun atur, dinamakan SGLL. Salah satu ciri utama SGLL adalah penggunaan penghubung, yang merupakan satu konsep yang digunakan dalam bahasa pengaturcaraan fungsian. Kami merasakan bahawa penghubung merupakan satu konsep yang lebih senang difahami berbanding pendekatan yang digunakan oleh Java, yang melibatkan penambahan alat kawalan kepada sebuah pengurus susun atur. Tambahan pula, kami percaya bahawa penyingkiran penyelerakan dan sokongan

yang lebih baik untuk pengabstrakan dalam pengendalian susun atur dapat meningkatkan produktiviti pengaturcara dan pemahaman kod sumber. Dalam tesis ini, kami memberikan tumpuan kepada kelas pengurus GridLayout, memandangkan kelas tersebut senang untuk difahami dan merupakan satu titik permulaan yang baik.

Untuk mengesahkan pendekatan kami, kami telah menilai Java dan SGLL dalam kedua-dua produktiviti dan pemahaman. Kami mendapati bahawa hasil kami menunjukkan bahawa SGLL dapat meningkatkan produktiviti dan pemahaman untuk tugas pengendalian susun atur.

ACKNOWLEDGEMENTS

Praise be to Allah for giving me the strength and wisdom to complete this work.

Without Allah's help and blessing, I would not have succeeded.

Firstly, I would like to thank my supervisor, Prof. Madya Dr. Abdul Azim Abd. Ghani, Dean of the Faculty of Computer Science and Information Technology, for invaluable discussion and commentary. Also, I would like to thank my co-supervisor, Prof. Madya Dr. Hj. Md. Nasir Hj. Sulaiman, for his support and encouragement.

I would also like to express my thanks to the Faculty of Computer Science and Technology, especially the ICT unit, for providing general help and assistance for the experiments. Also, I'd like to thank the Library and the School of Graduate Studies for helpfully fulfilling my every request.

Special thanks to my friends and colleagues at the Faculty of Computer Science and Information Technology for their support and advice. Your help will not be forgotten.

Finally, I would like to thank my family for giving me the motivation and moral support needed to complete this thesis. Only Allah can truly reward what they have done.

Khairul Azhar Kasmiran

March 2006



I certify that an Examination Committee has met on 17 March 2006 to conduct the final examination of Khairul Azhar Kasmiran on his Master of Science thesis entitled "Graphical User Interface Layout Language Using Combinators" in accordance with Universiti Pertanian Malaysia (Higher Degree) Act 1980 and Universiti Pertanian Malaysia (Higher Degree) Regulations 1981. The Committee recommends that the candidate be awarded the relevant degree. Members of the Examination Committee are as follows:

RAMLAN MAHMUD, PhD

Associate Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Chairman)

FATIMAH AHMAD, PhD


Associate Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Internal Examiner)

RUSLI ABDULLAH, PhD

Lecturer
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Internal Examiner)

MD. YAZID MOHD. SAMAN, PhD

Professor
Faculty of Science and Technology
Kolej Universiti Sains dan Teknologi Malaysia
(External Examiner)



HASANAH MOHD. GHAZALI, PhD
Professor/Deputy Dean
School of Graduate Studies
Universiti Putra Malaysia

Date: **11 JUL 2006**

This thesis submitted to the Senate of Universiti Putra Malaysia and has been accepted as fulfilment of the requirement for the degree of Master of Science. The members of the Supervisory Committee are as follows:

ABDUL AZIM ABD. GHANI, PhD

Associate Professor

Faculty of Computer Science and Information Technology

Universiti Putra Malaysia

(Chairman)

MD. NASIR SULAIMAN, PhD

Associate Professor

Faculty of Computer Science and Information Technology

Universiti Putra Malaysia

(Member)

AINI IDERIS, PhD

Professor/Dean

School of Graduate Studies

Universiti Putra Malaysia

Date:



DECLARATION

I hereby declare that the thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UPM or other institutions.



KHAIRUL AZHAR KASMIRAN

Date: 13/09/2006

TABLE OF CONTENTS

	Page
ABSTRACT	ii
ABSTRAK	iv
ACKNOWLEDGEMENTS	vi
APPROVAL	vii
DECLARATION	ix
LIST OF TABLES	xii
LIST OF FIGURES	xiii
LIST OF ABBREVIATIONS	xv
CHAPTER	
1 INTRODUCTION	1
1.1 Background	1
1.2 Problem Statement	3
1.3 Objective of the Research	4
1.4 Scope of the Research	4
1.5 Thesis Outline	5
2 LITERATURE REVIEW	8
2.1 Introduction	8
2.2 Graphical User Interface Terms and Definitions	8
2.3 Domain-specific Languages	10
2.4 Programming Language Attributes	11
2.4.1 General Language Criteria	11
2.4.2 Productivity (Writability)	15
2.4.3 Understandability (Readability)	16
2.5 Layout Handling	17
2.5.1 Functional Languages	17
2.5.2 Declarative Languages	21
2.5.3 Imperative Languages	23
2.6 Java and Swing	24
2.7 Evaluation of Programming Languages	26
2.8 Summary	28
3 METHODOLOGY	31
3.1 Introduction	31
3.2 Design of SGLL	31
3.2.1 Syntax of SGLL	32
3.2.2 Semantics of SGLL	33
3.3 Compiler Implementation Methodology	34
3.4 Experimental Design for the Productivity Experiment	35
3.4.1 Productivity Experiment Implementation	38
3.4.2 Confounding Factors for the Productivity Experiment	39
3.5 Experimental Design for the Understandability Experiment	41
3.5.1 Understandability Experiment Implementation	46
3.5.2 Confounding Factors for the Understandability Experiment	46



3.6	Summary	47
4	IMPLEMENTATION	48
4.1	Introduction	48
4.2	Syntax of SGLL (Support for Abstraction)	48
4.2.1	Support for the Control Hierarchy	49
4.2.2	Combinators	51
4.2.3	Support for Control Properties	53
4.3	Semantics of SGLL	54
4.3.1	Semantic Equivalence	54
4.3.2	Abstract Syntactic Domains	55
4.3.3	Abstract Production Rules	55
4.3.4	Semantic Domains	56
4.3.5	Semantic Functions	59
4.3.6	Semantic Equations	61
4.3.7	Auxiliary Functions	62
4.3.8	Comments	64
4.4	Compiler Implementation	65
4.4.1	Steps Taken in Building the Compiler	70
4.5	Summary	74
5	ANALYSIS AND DISCUSSION OF RESULTS	75
5.1	Introduction	75
5.2	Productivity Experiment	75
5.3	Understandability Experiment	79
5.4	Summary	83
6	CONCLUSION AND FUTURE WORK	84
6.1	Introduction	84
6.2	Contributions of the Research	84
6.3	Future Work	85
6.3.1	Additional Features	85
6.3.2	Further Evaluation	85
6.3.3	Tool Development	86
6.4	Conclusion	87
	REFERENCES	88
	APPENDICES	91
	BIODATA OF THE AUTHOR	127

LIST OF TABLES

Table	Page
2.1 List of language criteria from Appleby and VandeKopple (1997)	12
3.1 Experimental design for the productivity experiment	35
3.2 Experimental design for the understandability experiment	41
5.1 Data from the Java productivity experiment	76
5.2 Data from the SGLL productivity experiment	77
5.3 Raw data for the ANOVA test for productivity (units are seconds)	78
5.4 Data from the Java understandability experiment	80
5.5 Data from the SGLL understandability experiment	81
5.6 Raw data for the Friedman test for understandability (units are percentages)	82

LIST OF FIGURES

Figure	Page
1.1 Layout handling pseudocode for Java	3
2.1 Layout produced by TkGofer for “(A << B) ^ C”	20
2.2 Layout produced by FranTk for “title 'beside' timevalue”	20
2.3 Example of GroovyMarkup code	22
2.4 Example of Eve code	23
2.5 Example of Tk code	24
2.6 Example of Java code	25
4.1 A sample layout	49
4.2 Partitioning of the layout in Figure 4.1	49
4.3 SGLL code for the layout in Figure 4.2	49
4.4 Java code for the layout in Figure 4.2	50
4.5 Separated containers version of Figure 4.3	51
4.6 The 'above' combinator	52
4.7 The 'beside' combinator	52
4.8 Setting properties in Java	53
4.9 Setting properties in SGLL	53
4.10 Overall structure of the SGLL compiler	66
4.11 Example SGLL layout	67
4.12 Layout trees for SGLL layout in Figure 4.11	67
4.13 A layout with no children	68
4.14 Tree for layout in Figure 4.13	68
4.15 Control table structure	69
4.16 Smallest SGLL program	70

4.17 Example of a one-control program	70
4.18 Example of an n -control, one combinator program	71
4.19 Example of an n -combinator (single type) program	71
4.20 Example of a nested containers + y -axis combinator program	72
4.21 Example of single/multi-line comments	73
4.22 Example of a separated containers program	73



LIST OF ABBREVIATIONS

DSL	Domain-specific Language
GRAIL	Genuinely Readable And Intuitive Language
GUI	Graphical User Interface
IDE	Integrated Development Environment
MSDOS	Microsoft Disk Operating System
SGLL	Swing GUI Layout Language

CHAPTER 1

INTRODUCTION

1.1 Background

In 1995, Sun Microsystems presented the Java (Sun Microsystems, 2005) language to the world. Nowadays, it is one of the most popular programming languages, as evidenced informally by the TIOBE Programming Community Index (Tiobe Software, 2005).

Java is an object-oriented imperative language, i.e., it has support for objects as an abstraction method, as well as being based on the von Neumann architecture, with its emphasis towards variables, assignment statements and the iterative form of repetition. Imperative programming has a long history, starting with FORTRAN in the 1950s and continuing on with languages such as C (in the 1970s) and Java (Sebesta, 1999).

Since Java is a general-purpose programming language, its syntax will not be heavily tailored towards any specific programming task. Therefore, there are cases where a domain-specific language (or special-purpose language), tailored for a certain task, is more appropriate for the job. Such languages can provide significant boosts in productivity, usually by allowing the programmer to write less code for the same results (Spinellis and Guruprasad, 1997).

One area, in which we consider Java syntax to be deficient to the task at hand, is for the task of layout handling. A *layout* specifies the relative position and size of controls in an application's Graphical User Interface (GUI). While it is certainly possible to write layout handling code in Java itself, we suggest that it would be better for the programmer if a domain-specific language is used instead, since a domain-specific language can provide greater expressive power (van Deursen, *et al.*, 2000).

Thereby, we present a new domain-specific language based on Java that is designed for layout handling, named Swing GUI Layout Language or SGLL (Swing is the official GUI toolkit for Java). We present evidence in this thesis that our language provides significant boosts in programmer productivity and source code understandability compared to Java.

In this thesis, we define *productivity* as the time taken to perform a specific layout-handling task. We also define *understandability* generally as the ease in which programs can be read and understood, and empirically as the percentage of the answer containing errors when converting a layout in source code form to a layout in graphical form, given a fixed amount of time.

The compiler for SGLL is a source code generator. The generator produces Java code from a specification written in SGLL. The Java code produced can be compiled and executed like a normal Java program. The SGLL language itself is declarative. We represent a layout using a data structure, rather than as computer instructions.

There has been work done in the literature using a source code generator with initial results that show development of a complete user interface using that generator to be faster than using raw Java (da Silva, *et al.*, 2000). In this study, instead of developing a source code generator that can produce a complete user interface in Java, we concentrate only on the layout manager approach used by Java, since comparison of individual language features provide the most scientific results (Furuta and Kemp, 1979).

1.2 Problem Statement

Currently, Java uses the *layout manager* approach for layout handling. In the layout manager approach, each *container* (or parent for a group of controls) has a layout manager associated with it. The task of this layout manager is to arrange the controls under its supervision using a certain set of rules, specific to each layout manager.

Figure 1.1 shows the general structure of layout handling pseudocode for Java.

```
container.setLayout(new LayoutManager(initial parameters));  
container.add(control1, layout parameters);  
container.add(control2, layout parameters);  
container.add(control3, layout parameters);
```

Figure 1.1: Layout handling pseudocode for Java

Therefore, we have identified the following problems with the current approach:

- The layout resulting from that code is not apparent from the code itself. More specifically, the linear adding of controls does not correspond well to the tree structure of a layout hierarchy, thus reducing understandability.

- There is redundancy in the Java approach. More specifically, the programmer needs to type “*container.add*” every time he or she adds a new control to a container. Not only does such redundancy reduce productivity by adding the programmer’s workload unnecessarily, but such redundancy also reduces understandability by cluttering up the source code.

1.3 Objective of the Research

The objective of this research is to design and implement a GUI layout language for Swing. During language design, emphasis will be given on the language attributes of productivity and understandability. For productivity, the measurement to be used is the time taken to produce a layout, using either the Java or SGLL programming language as appropriate. For understandability, the measurement to be used is the percentage of the answer containing errors. The answer here refers to a hand-drawn layout that is directly derived from Java or SGLL source code as appropriate.

Our definitions of productivity and understandability are equivalent to Sebesta’s definitions of writability and readability respectively (Sebesta, 1999). We have chosen the two attributes for evaluation in this study because, according to Sebesta (1999), they are considered to be the most important attributes for language users.

1.4 Scope of the Research

This research is scoped according to the following delimitations: