Recognition of major contributors of an Ecma Standard / Technical Report (Past Ecma practice)

Source: I. Sebestyen

Background:

From the GA/2012/069 June 2012 GA Minutes:

> *"Then – on the question of the Editor of the ECMAScript Internationalization standard - there was a discussion if in an Ecma standard major contributing people could be listed on the informal part of the standard if they wanted to be, or not. It has been suggested that we should look into the past and current practice of Ecma and other SDOs on the matter.* ***It was then decided that the CC should investigate the matter: a) concrete recognition of people listed in the Ecma standard / technical report and b) a general way of recognition of key experts in an Ecma standard / technical report.*** *We should both look what are the benefits for Ecma and what for the members of such a recognition."*

Findings:

The Ecma Secretariat has looked into the matter and has found several past practices in Ecma. Some of them are incorporated below.

The only common feature is that they are all in the informative parts of the standards / TR.

Where fast-track to ISO was done, some were taken over by ISO unchanged, others were left out. E.g. in case of ECMA-357 the list of contributing people were left out from the ISO standard, but in case of ECMA-334 or ECMA-367 it was fully taken over.

There are also cases when only one individual is mentioned, but not a team of contributors. Sometimes as key contributor, in other cases as dedication to a person.

There are cases when only the name of contributing companies are mentioned (we should discuss that case separately).

There is no formal regulation on this in By-laws, Rules etc.

Proposals:

- If TCs request Ecma should permit the incorporation of names of contributors (persons) in the Ecma standard in the informative part of the standard. This is a simple way of giving credit to contributors and is done in several other SDO and Fora as well.
- Dedication to persons (e.g. who have passed away) should also be possible.
- We should list names of contributors (under their explicit consent when feasible), and/or if they so desire their affiliation.
- The above should be done in the "Introduction".

ECMA-357

Introduction

On 13 June 2002, a group of companies led by BEA Systems proposed a set of programming language extensions adding native XML support to ECMAScript (ECMA-262). The programming language extensions were designed to provide a simple, familiar, general purpose XML programming model that flattens the XML learning curve by leveraging the existing skills and knowledge of one of the largest developer communities worldwide. The benefits of this XML programming model include reduced code complexity, tighter revision cycles, faster time to market, decreased XML footprint requirements and looser coupling between code and XML data.

The ECMAScript group (Ecma TC39-TG1) unanimously agreed to the proposal and established a sub-group to standardize the syntax and semantics of a general purpose, cross platform, vendor neutral set of programming language extensions called ECMAScript for XML (E4X). The development of this Standard started on 8 August 2002. This Standard was developed as an extension to ECMAScript Edition 3, but may be applied to other versions of ECMAScript as well.

This Standard adds native XML datatypes to the ECMAScript language, extends the semantics of familiar ECMAScript operators for manipulating XML data and adds a small set of new operators for common XML operations, such as searching and filtering. It also adds support for XML literals, namespaces, qualified names and other mechanisms to facilitate XML processing.

This Standard will be integrated into future editions of ECMA-262 (ECMAScript). The ECMAScript group is working on significant enhancements for future editions of the ECMAScript language, including mechanisms for defining XML types using the XML Schema language and support for classes.

The following people have contributed to this specification:

John Schneider, BEA/AgileDelta (Lead Editor)
Rok Yu, Microsoft (Supporting Editor)
Jeff Dyer, Macromedia (Supporting Editor)

Steve Adamski, AOL/Netscape
Patrick Beard, AOL/Netscape
Adam Bosworth, BEA
Steve Brandli, BEA
Vikram Dhaneshwar, Microsoft
Brendan Eich, Mozilla Foundation
Vera Fleischer, Macromedia
Nathanial Frietas, palmOne
Gary Grossman, Macromedia
Waldemar Horwat, AOL/Netscape
Ethan Hugg, AgileDelta
Mark Igra, BEA
David Jacobs, MITRE
Alex Khesin, BEA
Terry Lucas, BEA
Milen Nankov, AgileDelta
Brent Noorda, Openwave
Richard Rollman, AgileDelta
Markus Scherer, IBM
Werner Sharp, Macromedia
Michael Shenfield, RIM
Edwin Smith, Macromedia
Dan Suciu, University of Washington
Peter Torr, Microsoft
Eric Vasilik, BEA
Herman Venter, Microsoft
Wayne Vicknair, IBM
Roger Weber, BEA

This Ecma Standard has been adopted by the General Assembly of December 2005.

(ECMA-367)

Introduction

Eiffel was originally designed in 1985 as a method of software construction and a notation to support that method The first implementation, from Eiffel Software (then Interactive Software Engineering Inc.), was commercially released in 1986. The principal designer of the first versions of the language was Bertrand Meyer. Other people closely involved with the original definition included Jean-Marc Nerson. The language was originally described in Eiffel Software technical documents that were expanded to yield Meyer's book *Eiffel: The Language* in 1990-1991. The two editions of *Object-Oriented Software Construction* (1988 and 1997) also served to describe the concepts. (For bibliographical references on the documents cited see Clause 3) As usage of Eiffel grew, other Eiffel implementations appeared, including Eiffel/S and Visual Eiffel from Object Tools, Germany, EiffelStudio and Eiffel Envision from Eiffel Software, and SmartEiffel from LORIA, France.

Eiffel today is used throughout the world for industrial applications in banking and finance, defense and aerospace, health care, networking and telecommunications, computer-aided design, game programming, and many other application areas. Eiffel is particularly suited for mission-critical developments in which programmer productivity and product quality are essential. In addition, Eiffel is a popular medium for teaching programming and software engineering in universities.

In 2002 Ecma International formed Technical Group 4 (Eiffel) of Technical Committee 39 (Programming and Scripting Languages). The *Eiffel: Analysis, Design and Programming Language* Standard provides a precise definition of the language and ensures interoperability between implementations. The first of these benefits is of particular interest to implementors of Eiffel compilers and environments, who can rely on it as the reference on which to base their work; the second benefit is to Eiffel users, for whom the Standard delivers a guarantee of compatibility between the products of different providers and of trust in the future of Eiffel.

TG4 devised this Standard from June 2002 to April 2005, starting from material from the original and revised versions of the book *Standard Eiffel* (latest revision of *Eiffel: The Language*). During that period, the Technical Group conducted fifteen face-to-face meetings and numerous phone meetings, in addition to extensive technical correspondence. The members of the committee have been: Karine Arnout (ETH, Zurich); Éric Bezault (Axa Rosenberg, Orinda); Paul Cohen (Generic, Stockholm), Dominique Colnet (LORIA, Nancy); Mark Howard (Axa Rosenberg, Orinda); Alexander Kogtenkov (Eiffel Software, Moscow); Bertrand Meyer (Eiffel Software, Santa Barbara, and ETH, Zurich); Christine Mingins (Monash University, Melbourne); Roger Osmond (EMC, Boston); Emmanuel Stapf (Eiffel Software, Santa Barbara); Kim Waldén (Generic, Stockholm).

Observers having attended one or more of the meetings include: Cyril Adrian (LORIA), Volkan Arslan (ETH), Paul Crismer (Groupe S, Brussels), Jocelyn Fiat (Eiffel Software, France), Randy John (Axa Rosenberg), Ian King (Eiffel Software), Philippe Ribet (LORIA), Julian Rogers (Eiffel Software), Bernd Schoeller (ETH), David Schwartz (Axa Rosenberg), Zoran Simic (Axa Rosenberg), Raphael Simon (Eiffel Software), Olivier Zendra (LORIA). The committee acknowledges the contributions of many people, including David Hollenberg, Marcel Satchell, Richard O'Keefe and numerous others listed in the acknowledgments of the book *Standard Eiffel*.

The editor of the Standard is Bertrand Meyer. Emmanuel Stapf is the convener of TG4 (succeeding Christine Mingins, 2002-2003). Karine Arnout is the recording secretary of TG4.

The final version of the document was prepared by Éric Bezault, Mark Howard, Alexander Kogtenkov, Bertrand Meyer and Emmanuel Stapf.

© ISO/IEC 2006 – All rights reserved

(ECMA-334)

Introduction

This International Standard is based on a submission from Hewlett-Packard, Intel and Microsoft, that described a language called C#, which was developed within Microsoft. The principal inventors of this language were Anders Hejlsberg, Scott Wiltamuth and Peter Golde. The first widely distributed implementation of C# was released by Microsoft in July 2000, as part of its .NET Framework initiative.

Ecma Technical Committee 39 (TC39) Task Group 2 (TG2) was formed in September 2000, to produce a standard for C#. Another Task Group, TG3, was also formed at that time to produce a standard for a library and execution environment called Common Language Infrastructure (CLI). (CLI is based on a subset of the .NET Framework.) Although Microsoft's implementation of C# relies on CLI for library and runtime support, other implementations of C# need not, provided they support an alternate way of getting at the minimum CLI features required by this C# standard (see Annex D).

As the definition of C# evolved, the goals used in its design were as follows:

- C# is intended to be a simple, modern, general-purpose, object-oriented programming language.
- The language and implementations thereof should provide support for software engineering principles such as strong type checking, array bounds checking, detection of attempts to use uninitialized variables, and automatic garbage collection. Software robustness, durability and programmer productivity are important.
- The language is intended for use in developing software components suitable for deployment in distributed environments.
- Source code portability is very important, as is programmer portability, especially for those programmers already familiar with C and C++.
- Support for internationalization is very important.
- C# is intended to be suitable for writing applications for both hosted and embedded systems, ranging from the very large that use sophisticated operating systems, down to the very small having dedicated functions.
- Although C# applications are intended to be economical with regard to memory and processing power requirements, the language was not intended to compete directly on performance and size with C or assembly language.

The following companies and organizations have participated in the development of this International Standard, and their contributions are gratefully acknowledged: ActiveState, Borland, CSK Corp., Hewlett-Packard, IBM, Intel, IT University of Copenhagen, Jaggersoft (UK), Microsoft, Mountain View Compiler, Monash University (AUS), Netscape, Novell, Pixo, Plum Hall, Sun, and the University of Canterbury (NZ).

The development of this version of the International Standard started in January 2003.

©ISO/IEC 2006 – All rights reserved

ECMA-217

Brief History

This Standard defines Phase II of Services for Computer Supported Telecommunications Applications (CSTA) for OSI Layer 7 communication between a computing network and a telecommunications network. This Standard and its companion Standard ECMA-218 ***Protocol for Computer Supported Telecommunications Applications (CSTA) Phase II*** reflect agreements of ECMA member companies on Phase II of the standards for CSTA. Additional phases are anticipated. This Standard is based on the practical experience of ECMA member companies and represents a pragmatic and widely-based consensus.

This Standard takes direction from Technical Report ECMA TR/52 ***Computer Supported Telecommunications Applications.***

Phase II of CSTA extends the previous Phase I standard in major theme directions as well as numerous details. Major areas of advancement include:

- the addition of explicit application context negotiation mechanisms;
- the addition of I/O services;
- the addition of Special Resource Functions and, particularly, Voice Unit services;
- new and/or enhanced services and event reports for commonly used call control and monitoring applications; new services include Single Step Transfer, Single Step Conference, Call Park and Send DTMF Signals.

The Phase II CSTA standards are not fully backwards compatible with the Phase I standards. Although backwards compatibility is an important consideration and has been maintained whenever possible, the addition of new parameters in certain services and events, as well as the deletion of unused Phase I services and the addition of entirely new Phase II services and events, did not allow complete backwards compatibility.

This Standard is dedicated to the memory of Terry Wuerfel.

This ECMA Standard has been adopted by the ECMA General Assembly of December 1994.

ECMA TR/68

Brief History

This Technical Report is based upon Phase II of Services for Computer Supported Telecommunications Applications (CSTA) and introduces CSTA Scenarios. This Technical Report uses Standards ECMA-217 ***Services for Computer Supported Telecommunications Applications Phase II*** and ECMA-218 ***Protocol for Computer Supported Telecommunications Applications Phase II*** to illustrate how CSTA services and events may be used in typical call scenarios. It reflects a common understanding of ECMA member companies. Additional phases of this Technical Report are anticipated. This Technical Report is based on the practical experience of ECMA member companies and represents a pragmatic and widely-based consensus.

The Phase II CSTA standards are not fully backwards compatible with the Phase I standards. Although backwards compatibility is an important consideration and has been maintained whenever possible, the addition of new parameters in certain services and events, as well as the deletion of unused Phase I services and the addition of entirely new Phase II services and events, did not allow complete backwards compatibility.

This Technical Report is dedicated to the memory of Terry Wuerfel

This ECMA Technical Report has been adopted by the ECMA General Assembly of December 1994.

TR98

Introduction

This document specifies a file format, referred to as the JPEG File Interchange Format (JFIF), for file-based interchange of images encoded according to the JPEG standard (ITU-T Recommendation T.81 | ISO/IEC 10918-1).

The JPEG File Interchange Format (JFIF) was collaboratively developed by a group of computer, telecommunications, and imaging companies in the early 1990's. Representing C-Cube Microsystems, Eric Hamilton led the development of the specification. He hosted a meeting in late 1991 toward developing a simple file format based on JPEG which would allow for the interchange of files containing JPEG bitstreams between platforms and applications. There were about 40 representatives from various computer, telecommunications, and imaging companies at the meeting. Subsequent specification development work was conducted using e-mail and telephone discussions. The effort reached consensus fairly quickly and led to publication of version 1.0 of the JFIF specification, which had been edited by Eric Hamilton and which he distributed to the participants and to other interested parties.

Shortly thereafter, the group came to the conclusion that the spatial sampling relationship of components specified in JFIF 1.0 was not ideal since it followed digital video conventions rather than those used in common computer formats such as Postscript and QuickTime. They chose to publish another version, JFIF 1.01, which changed that part of the specification to follow the computer format convention. This was deemed to be a minor change since JFIF 1.0 had been circulated for only a short while and decoders which ignored the version number would still render similar images. The Independent JPEG Group (IJG) adopted JFIF version 1.01 for use in its public domain software, which eventually led to millions of images being published in this format. Later in 1992, user feedback led to the final version of JFIF, version 1.02, which supported additional thumbnail formats – most importantly including thumbnails stored in compressed format.

The JFIF Version 1.02 specification became available in a *de facto* manner in the public domain, and it has been implemented widely, to the extent that it became widely recognized as a *de facto* standard.

The desire to convert JFIF Version 1.02 to a formal publication status has been a topic of various discussions since its development, but no action was taken on that subject until preparation of this specification. The rationale for this Ecma International Technical Report is therefore to formally document JFIF Version 1.02 in a technically identical Ecma International Technical Report.

By preparing this specification, Ecma International acknowledges and applauds the excellent work that Eric Hamilton and the JFIF group did when creating the JFIF specification, and expresses its gratitude to the supporting organizations, including in particular C-Cube Microsystems (now LSI Logic) and the other companies that have participated in the original JFIF work.

Ecma International also expresses its sincere gratitude to Eric Hamilton for his valuable help in editing this specification.

This Ecma Technical Report has been adopted by the General Assembly of June 2009.

© Ecma International 2009 – All rights reserved