Fourth-generation programming language

From Wikipedia, the free encyclopedia

A **fourth-generation programming language** (**4GL**) is a computer programming language envisioned as a refinement of the style of languages classified as third-generation programming language (3GL). Each of the programming language generations aims to provide a higher level of abstraction of the internal computer hardware details, making the language more programmer-friendly, powerful and versatile. While the definition of 3GL has changed over time, it can be typified by operating more with large collections of information at once rather than focusing on just bits and bytes. Languages claimed to be 4GL may include support for database management, report generation, mathematical optimization, GUI development, or web development. Some researchers state that 4GLs are a subset of domain-specific languages.[1][2]

The concept of 4GL was developed from the 1970s through the 1990s, overlapping most of the development of 3GL. While 3GLs like C, C++, C#, Java, and JavaScript remain popular for a wide variety of uses, 4GLs as originally defined found narrower uses. Some advanced 3GLs like Python, Ruby, and Perl combine some 4GL abilities within a general-purpose 3GL environment. Also, libraries with 4GL-like features have been developed as add-ons for most popular 3GLs. This has blurred the distinction of 4GL and 3GL.

In the 1980s and 1990s, there were efforts to develop fifth-generation programming languages (5GL).

Contents

History

Though used earlier in papers and discussions, the term 4GL was first used formally by James Martin in his 1982 book *Applications Development Without Programmers*[3] to refer to non-procedural, high-level specification languages. In some primitive way, early 4GLs were included in the Informatics MARK-IV (1967) product and Sperry's MAPPER (1969 internal use, 1979 release).

The motivations for the '4GL' inception and continued interest are several. The term can apply to a large set of software products. It can also apply to an approach that looks for greater semantic properties and implementation power. Just as the 3GL offered greater power to the programmer, so too did the 4GL open up the development environment to a wider population.

In a sense, the 4GL is an example of 'black box' processing, each generation (in the sense of the page) is further from the machine (see the Computer Science history in regard to data structure improvements and information hiding). It is this latter nature that is directly associated with 4GL having errors that are harder, in many cases, to debug. In terms of applications, a 4GL could be business oriented or it could deal with some technical domain. Being further from the machine implies being closer to domain. Given the wide disparity of concepts and methods across domains, 4GL limitations lead to recognition of the need for the 5GL.

The early input scheme for the 4GL supported entry of data within the 72-character limit of the punched card (8 bytes used for sequencing) where a card's tag would identify the type or function. With judicious use of a few cards, the 4GL deck could offer a wide variety of processing and reporting capability whereas the equivalent functionality coded in a 3GL could subsume, perhaps, a whole box or more of cards.[4]

The 72-character metaphor continued for a while as hardware progressed to larger memory and terminal interfaces. Even with its limitations, this approach supported highly sophisticated applications.

As interfaces improved and allowed longer statement lengths and grammar-driven input handling, greater power ensued. An example of this is described on the Nomad page.

> Another example of Nomad's power is illustrated by Nicholas Rawlings in his comments for the Computer History Museum about NCSS (see citation below). He reports that James Martin asked Rawlings for a Nomad solution to a standard problem Martin called the *Engineer's Problem*: "give 6% raises to engineers whose job ratings had an average of 7 or better." Martin provided a "dozen pages of COBOL, and then just a page or two of Mark IV, from Informatics." Rawlings offered the following single statement, performing a set-at-a-time operation...

The development of the 4GL was influenced by several factors, with the hardware and operating system constraints having a large weight. When the 4GL was first introduced, a disparate mix of hardware and operating systems mandated custom application development support that was specific to the system in order to ensure sales. One example is the MAPPER system developed by Sperry. Though it has roots back to the beginning, the system has proven successful in many applications and has been ported to modern platforms. The latest variant is embedded in the BIS[5] offering of Unisys. MARK-IV is now known as VISION:BUILDER and is offered by Computer Associates.

Santa Fe railroad used MAPPER to develop a system, in a project that was an early example of 4GL, rapid prototyping, and programming by users.[6] The idea was that it was easier to teach railroad experts to use MAPPER than to teach programmers the "intricacies of railroad operations".[7]

One of the early (and portable) languages that had 4GL properties was Ramis developed by Gerald C. Cohen at Mathematica, a mathematical software company. Cohen left Mathematica and founded Information Builders to create a similar reporting-oriented 4GL, called FOCUS.

Later 4GL types are tied to a database system and are far different from the earlier types in their use of techniques and resources that have resulted from the general improvement of computing with time.

- IBM Rational EGL (Enterprise Generation Language)
- LabVIEW
- LANSA
- LINC
- Omnis Studio SDK
- OutSystems (Productivity/PaaS)
- Panther
- PowerBuilder
- Perl
- PHP
- Python
- Ruby
- Oracle Application Development Framework
- DEC RALLY
- SheerPower4GL (Microsoft Windows only)
- SQLWindows/Team Developer
- uniDap GTD
- Uniface
- Visual DataFlex (Microsoft Windows only)
- WinDev
- Unix Shell
- XBase++
- Manage (SDS Sigma series mainframes)

Database query languages

- FOCUS
- 4D QL
- Informix-4GL
- OpenEdge ABL
- OpenROAD (Ingres 4GL)
- Ramis
- SQL

Report generators

- Tally Definition Language(TDL): This is domain specific language for ERP & Accounting from Tally solutions Pvt Ltd
- BuildProfessional
- LIRC
- NATURAL
- Oracle Reports
- Progress 4GL Query/Results
- Quest
- RPG-II

Data manipulation, analysis, and reporting languages

- Ab Initio
- ABAP
- Aubit-4GL
- Audit Command Language
- Clarion Programming Language
- CorVision
- Culprit
- ADS/Online (plus transaction processing)
- Easy[illegible]ev[illegible]
- GraphTalk
- IDL
- IGOR Pro
- Informix-4GL
- JSL
- LANSA
- LabVIEW
- MANTIS
- MAPPER (Unisys/Sperry) now part of BIS
- MARK-IV (Sterling/Informatics) now VISION:BUILDER of CA

- Simulink a component of MATLAB
- NATURAL
- Nomad
- PL/SQL
- Progress 4GL
- PROIV
- R
- Ramis
- S
- Scilab
- SAS

GUI creators

- 4th Dimension (Software)
- MATLAB's GUIDE
- Omnis Studio
- OpenROAD
- Progress 4GL AppBuilder
- SuperTalk

Mathematical optimization

- AIMMS
- AMPL
- GAMS

Database-driven GUI application development

- Action Request System
- C/AL
- Genexus
- SB+/SystemBuilder
- Progress Dynamics
- Unify VISION

Screen painters and generators

- SB+/SystemBuilder
- Oracle Forms

- SPSS
- SQL PL
- SQR
- Stata
- Synon
- Wolfram Language
- XBase++
- Xquery Backward compatible with SQL and forward compatible with XML data sources.

- Transcript (LiveCode)
- XUL Can be used with Xquery to create web GUI database applications quickly.
- Visual DataFlex

- Progress 4GL ProVision

Web development languages

- ActiveVFP
- CFML
- LANSA
- Wavemaker open source, browser-based development platform for Ajax development based on Dojo, Spring, Hibernate
- OutSystems

See also

- List of fourth-generation programming languages
- Domain-specific programming language
- Rapid application development
- Fifth-generation programming language

References

1. 35th Hawaii International Conference on System Sciences - 1002 Domain-Specific Languages for Software Engineering (http://csdl.computer.org/comp/proceedings/hicss/2002/1435/09/14350279.pdf&ei=pgcWQ6CwKsKYYMfF9OAI) Archived (https://web.archive.org/web/20110516121525/http://csdl.computer.org/comp/proceedings/hicss/2002/1435/09/14350279.pdf&ei=pgcWQ6CwKsKYYMfF9OAI) May 16, 2011, at the Wayback Machine.
2. Arie van Deursen; Paul Klint; Joost Visser (1998). "Domain-Specific Languages:An Annotated Bibliography". Archived from the original on 2009-02-02. Retrieved 2009-03-15.
3. Martin, James. *Application Development Without Programmers*. Prentice-Hall, 1981. ISBN 0-13-038943-9.
4. Columbia University Computing History: IBM Cards (http://www.columbia.edu/acis/history/cards.html)
5. Unisys. Business Information Server (http://www.unisys.com.hk/products/software/application__development/business__information__server/features.htm) (BIS).
6. Louis Schlueter, User-Designed Computing: The Next Generation, 1988. [book on report generator and MAPPER systems]
7. McNurlin & Sprague. Technologies for Developing Systems (http://telaga.cs.ui.ac.id/WebKuliah/IKI42400/2004/McNurlin-5ed-ch09.pdf) Information Systems Management in Practice. Prentice Hall, 2003. ISBN 0-13-101139-1

External links

- FourGen CASE Tools - Rapid Application Development Environment (http://www.gillani.com/CASETools.htm)
- Four J's Development Tools Genero, Genero Studio (http://www.4js.com)
- IBM Informix Genero (http://www.ibm.com/software/products/us/en/infogene/)
- 4GL GPL/GNU OpenSource development tools project (http://aubit4gl.sourceforge.net/)