INTEGRATED DEVELOPMENT ENVIRONMENT IN SHIPBUILDING COMPUTER SYSTEMS

A Benayas and A Cebollero, Sener Ingeniería y Sistemas SA, Spain

SUMMARY

Computer applications for shipbuilding were born to make ship design and production more efficient. The early programs soon became complex systems, expanding their functionality to all aspects of the ship life cycle. Eager to meet the ever-increasing requirements of their customers, software houses must provide not only advanced systems but also parallel development toolkits for users to develop in-house extensions.

Taking as a reference the Foran System, this paper reviews the key features of a class of those toolkits, the integrated development environments: intuitive programming language, insulation from mainstream system changes, reduced maintenance expense and rapid response to evolving requirements.

Our conclusion is that modern shipbuilding computer systems gain acceptance among designers and builders if they are able to offer alternative but convergent ways to develop in-house software extensions with reasonable effort, if only to perform repetitive tasks, integrate best-practice design rules, simplify design checks and automate drawing generation.

NOMENCLATURE

API Application Programming Interface
ECMA Ecma International, European association for standardizing information and communication systems
IDE Integrated Development Environment
PLM Product Life-Cycle Management
SDK Software Development Kit

1. INTRODUCTION

Computer aided ship design and related engineering and manufacturing applications were born to make design and production faster and more efficient. In pursuing these objectives, the early programs soon became complex systems, expanding their functionality to all aspects of the ship project life cycle. Users’ requirements also grew—and keep growing—rapidly, to the point that software houses are normally overwhelmed and typically face the dilemma of what they should develop to meet some especially demanding requirements: either new built-in commands or toolkits for users—or third-parties—to customize the common system.

In principle, it may seem to be only beneficial for users to have such development tools on hand, but we will see this is not always the best strategy. On the other hand, in many cases the availability of a toolkit looks like a dream and permits customization works that are otherwise unaffordable for the software vendor.

2. SOFTWARE DEVELOPMENT KITS

There exist today multiple options for end-users of shipbuilding computer systems to expand the out-of-the-box functionality. Their scope and possibilities vary largely, but in general we can say that all these options, which we would call collectively Software Development Kits (SDKs), share the purpose of facilitating the development work to people who are not necessarily IT experts, but are typically good at ship design or production and have brilliant ideas for enhancing the underlying system.

Basically, a SDK is a set of development tools that allows the creation of applications for a certain computer system or platform. It may be something as simple as an Application Programming Interface (API) or more sophisticated environments embedded in a system, including debugging aids and other advanced utilities.

Software vendors are opening their systems by providing one or more SDKs, in accordance with the most common programming standards and paradigms, in order to better meet the demand for in-house development platforms.

3. INTEGRATED DEVELOPMENT ENVIRONMENT

3.1 DEFINITION

Among the existing SDKs, this paper focuses on the Integrated Development Environment (IDE) concept.

All is in the name:
- Integrated: it forms integral part of a system, with which it interacts and shares resources.
- Development: it serves to create new applications, bringing the capabilities or possibilities of the mainstream system to a more effective state.
- Environment: it is a well-configured, comprehensive set of tools and resources conceived for creating professional extensions of the system.

An IDE normally consists of:
- Source Code Editor
- Compiler and/or Interpreter
- Automatic Builder
- Debugger
3.2 REQUIREMENTS

First requirement for an IDE is the ease of use. Above all, this is achieved by means of an intuitive programming language, ideally based on some international standard.

Nowadays, programming environments feature rich hierarchies of objects, properties, methods and event handlers that allow programmers to not only develop quickly but also reuse efficiently their code and schemas. Most environments are also web-oriented, offering the possibility of remote access and wide networking process.

These languages may be compiled or interpreted. For the same level of complexity of the source code, a compiled program tends to be much faster, but requires longer edit-link-run cycles. A combination of both approaches is increasingly common: a compiler translates the source code into some intermediate form –often called byte code–, which is then passed to an interpreter for execution.

As important as the language it is for an IDE the insulation from the mainstream system changes. It would make no sense to develop software extensions that become obsolete and impractical at the first update of the system. User-created commands and procedures should remain unchanged or easily –if not automatically– converted or adapted to the new system versions and releases.

Closely related to the previous requirement is the need for reduced maintenance expense, in both the IDE and the results obtained through it. The maintenance of an IDE requires a certain programming and documentation effort from software houses, in order to keep onward compatibility between system and existing user-created extensions. The use of international standards and languages of similar syntax in main system and IDE reduces significantly the maintenance effort.

Obviously, the easier to use and maintain the IDE is, the faster the end-user will be able to adapt the system to new challenges and improvement opportunities. Moreover, the availability of an IDE permits to balance the development workload between the vendor and the customer (or a third-party). This permits a quicker response to evolving functional requirements as well as keeping confidential know-how and procedures on the customer side.

3.3 LIMITS

Although the SDKs in general and the IDEs in particular seem to be the panacea to dealing with customer-specific requirements, it is not always the best way to address new developments, and in fact it is counter-indicated in multiple situations.

On the other hand, when customers largely base their design and production procedures on special in-house commands, the risk of misalignment with the substratum system grows considerably, and may even lead to irreversible incompatibility, which in turn results in increasing pressure on IT teams or excessive dependency on third-party companies.

4. AN ILLUSTRATIVE CASE: THE FORAN DEVELOPMENT ENVIRONMENT

4.1 FOUNDATION

Foran, the shipbuilding computer system developed by Sener, features a modern IDE –called FDE or Foran Development Environment–, which allows users and programmers to add new capabilities to the System.

Combining a scripting language fully compliant with the Ecma-262 standard (EcmaScript Language Specification) and Foran intrinsic functionality, FDE provides a strong engine for creating customised functions and commands.

On the other hand, when customers largely base their design and production procedures on special in-house commands, the risk of misalignment with the substratum system grows considerably, and may even lead to irreversible incompatibility, which in turn results in increasing pressure on IT teams or excessive dependency on third-party companies.

4. AN ILLUSTRATIVE CASE: THE FORAN DEVELOPMENT ENVIRONMENT

4.1 FOUNDATION

Foran, the shipbuilding computer system developed by Sener, features a modern IDE –called FDE or Foran Development Environment–, which allows users and programmers to add new capabilities to the System.

Combining a scripting language fully compliant with the Ecma-262 standard (EcmaScript Language Specification) and Foran intrinsic functionality, FDE provides a strong engine for creating customised functions and commands.

FDE integrates world-class commercial components, provided by Nokia Qt Development Frameworks (formerly Trolltech), and Sener-developed software:
- Customised embedded Editor based on QtScripting
- Interpreter (QtScript)
- Foran Wrappers
- Auto-Builder
- Integrated Debugger

Figure 1: A view of the FDE Workbench

Figure 2: FDE Architecture
FDE uses QtScripting wrappers to access the Foran application frameworks. These wrappers form an intermediate layer that glues the programming environment with the necessary Foran objects and maintains System’s coherence in a transparent way. Wrappers cover both general tasks (dialogs, database access, document generation/edition, file handling, …) and design-specific tasks (3D scene, element properties, part generation, XML input/output interfaces, …).

Figure 3: FDE Modules

FDE fully supports Unicode, so users should not run into any problems if they are using an international character set. It also provides easy-to-use macro recording and execution functionalities.

4.2 EXAMPLES OF APPLICATION

FDE was conceived keeping in mind the ease of use in order to be practical and efficient for system developers and designers of different levels.

Typical tasks that can be done through FDE applications include:
- Access to product model database
- Fully customised reports with graphics
- Implementation of rule-based design capabilities
- Automation of processes frequently repeated
- Integration with other systems

A good example of FDE use is the design comparison to highlight in the 3D scene modifications made to the project by a given user, between certain dates, affecting some particular user-defined attributes or as the result of a combination of those variables. This extended function was made by programming a sequence of interactive filters for the different data types and a specific search process of matching elements, which can be visualized in a distinctive colour.

Figure 4: FDE application to highlighting model changes

The elaboration of production drawings is one of the most time-consuming tasks in the ship design cycle, and here is where CAD systems can be, and in fact are, of the greatest help to designers, making drawings as much as possible automatically. Yet these drawings must also adhere to shipyard-specific rules and styles, therefore their customization is a critical requirement. At this point FDE is particularly welcome, as it permits the highest level of drawing customisation, preview and selection on top of Foran’s built-in capability to automatically generate 2D drawings tied to the 3D product model.

Figure 5: FDE-customised, automatically generated pipe support drawing

FDE serves not only for end-users to customise their Foran installations, but also for Sener programmers to improve, in certain cases, the main system. For instance, this has happen to the new ‘Search in 3D Scene’ command, which has been improved in the latest release by adding a previously created FDE command for filtering 3D items by user-defined attributes.

Figure 6: New Search command, with an FDE touch

In much the same way, many Foran commands are also developed with FDE, especially in Outfitting and Electrical Design modules.
5. CONCLUSIONS

Modern shipbuilding computer systems gain acceptance among designers and builders when they offer alternative but convergent ways to develop in-house software extensions with reasonable effort and easy maintenance, if only to perform repetitive tasks, integrate best-practice design rules, simplify design checks and automate drawing completion.

IDEs are among the most practical SDKs, especially when based on international standards. In particular, Foran [Integrated] Development Environment has proven to be a powerful, reliable, programmer- and user-friendly tool set for expanding and enhancing the System capabilities, when used in the right circumstances and conditions.

6. AUTHORS’ BIOGRAPHIES

Arturo Benayas, MSc Naval Architecture & Marine Engineering from Polytechnic University of Madrid, has been a member of the Foran Customer Support Group at Sener Ingeniería y Sistemas SA since 2005, with particular responsibility for technical support and programming FDE applications on demand.

Alfonso Cebollero, MSc Naval Architecture & Marine Engineering from Polytechnic University of Madrid and MA Shipping Business from ICADE, currently holds the position of Regional Director for Asia & Pacific in the Marine Systems Department of Sener Ingeniería y Sistemas SA. He joined Sener in 1994 after having worked several years for another engineering company, a shipping consultancy and a naval shipyard.