Technologie projektowania GIS i możliwości narzędzi projektowych

DROGA DO SYSTEMU

Technologie oraz zintegrowane środowiska programowe ułatwiają tworzenie systemów informatycznych, które mogą mieć formę rozbudowaną (jak wielofunkcyjny system geodezyjny) bądź też wykonywać konkretne czynności (np. wymianę danych przestrzennych między dwoma systemami).

BEATA ŻERO

System informatyczny tworzą powiązane ze sobą elementy, takie jak dane, zasoby techniczne i ludzkie. Ta definicja jest rozbudowywana w zależności od potrzeb konkretnej jednostki oraz realizowanych przez nią zadań. Mówimy wówczas o systemach dedykowanych, czego przykładem może być system informacji geograficznej (GIS). Zadaniem GIS jest pozyskiwanie, przetwarzanie i udostępnianie danych zawierających informacje przestrzenne oraz towarzyszące im informacje opisowe o obiektach wyróżnionych w części przestrzeni objętej działaniem systemu [Gaździcki, 1990].

TECHNOLOGIA BUDOWY SYSTEMU

Struktura wszystkich systemów informatycznych jest jednakowa. Konstrukcja odnosi się do pewnych podstawowych celów i funkcji oraz zależy od tego, czy system będzie wystarczająco praktyczny, czy zaspokoi potrzeby użytkowników. Jeśli nie, to albo zostaje on ulepszony, albo wymieniony na inny, bardziej odpowiedni do planowanych zadań.

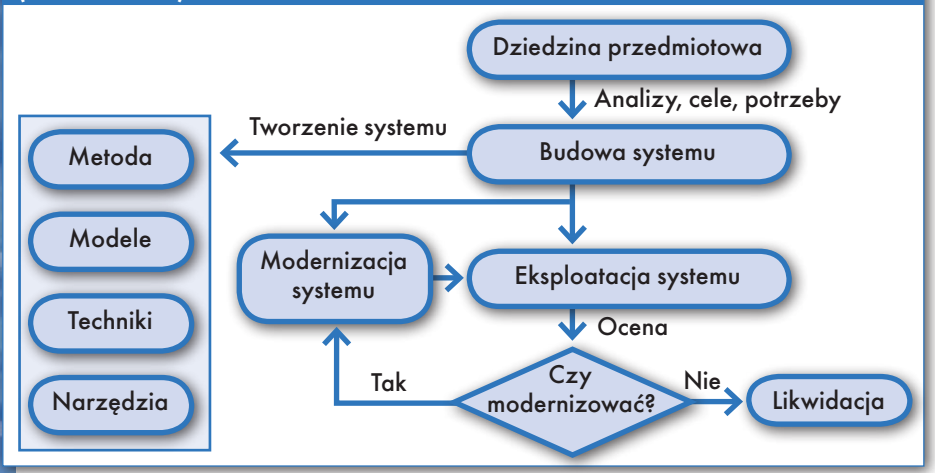
Zazwyczaj projektowany system jest złożony z wielu części, które mogą być samodzielnie działającymi aplikacjami. Mimo że działają one osobno, są powiązane z podstawowym modułem systemu, a często nawet – od niego zależne. Współpracują między sobą na podstawie pewnych ustalonych reguł i procedur.

Ogólnie można stwierdzić, że częściej modernizowane są wybrane fragmenty systemu niż jego podstawowy moduł. Udoskonalanie systemu polega również na tworzeniu nowych aplikacji spełniających określone czynności, zdefiniowane przez potrzeby wynikłe w trakcie pracy. Przykładem jest oprogramowanie firmy Geobid, gdzie podstawą jest EwMapa, aplikacją zaś – Sesut. Rozbudowanie funkcjonalności wiąże się z utworzeniem dodatkowego programu wspomagającego działania, np. eksport danych przestrzennych do formatu GML.

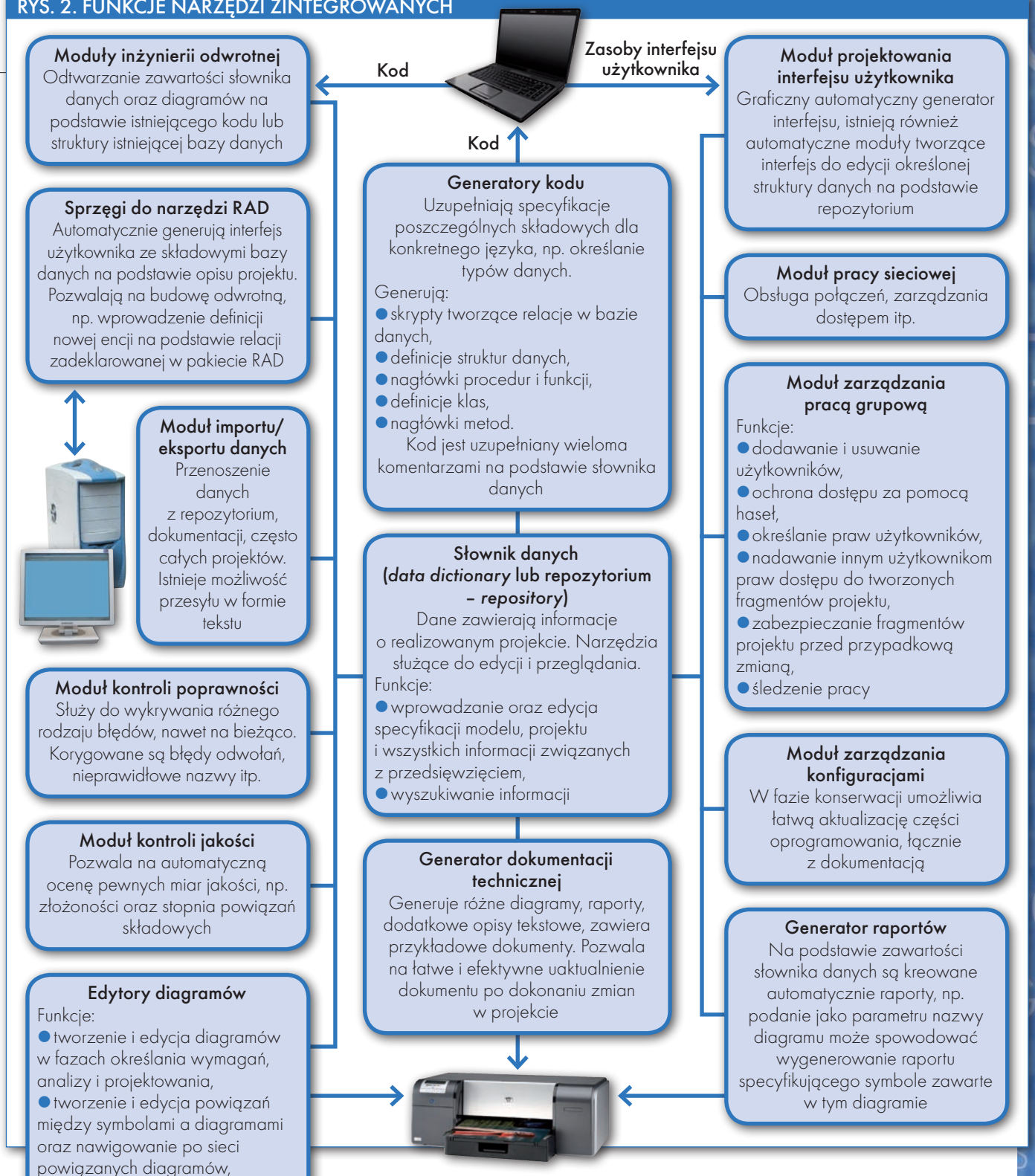
ETAPY BUDOWY SYSTEMU

Podstawową czynnością jest wybranie modelu, według którego będzie budowany system. Prace wykonywane w poszczególnych etapach zależą od konkretnego projektu, dla którego tworzony jest system. Przed przystąpieniem do realizacji zadania geodezyjnego należy przeprowadzić analizę zakresu planowanego programu. Mając zdefiniowane założenia, w kolejnym kroku należy dobrać technologię informatyczną budowy programu. Jak wiadomo, dzielą się one na obiektowe i strukturalne. To właśnie wybór technologii określa, jakie narzędzia i w jaki sposób będą użyte podczas realizacji projektu. Opis tych wielu metod, narzędzi i prac wykonywanych podczas realizacji poszczególnych etapów tworzenia syste-

RYS. 1. POWIĄZANIE SKŁADNIKÓW METODYKI I CYKLU ŻYCIA SYSTEMU (PROGRAMU) GIS



RYS. 2. FUNKCJE NARZĘDZI ZINTEGROWANYCH



mu można znaleźć w literaturze z zakresu inżynierii programowania. Niełatwe jest wybranie optymalnej ścieżki budowy programu, aby prace przebiegały płynnie. Etapy prac w znacznym stopniu ulegają uproszczeniu, jeśli projektant jest jednocześnie programistą, ale możliwe jest to jedynie przy niewielkich projektach.

Dobór technologii jest dość trudnym zadaniem, jednak podstawą projektowania systemu jest wybranie właściwego cyklu wytwórczego. Budowa oprogramowania według podejścia strukturalnego może być oparta na kilku modelach często spotykanych w literaturze (np. kaskadowym, spiralnym), które nie wykluczają się wzajemnie. Opracowano również wiele metodyk, w których stosuje się podejście obiektowe. Opisują one możliwe realizacje procesu tworzenia systemu, np.: MDA (*Model Driven Architecture*), FDD (*Future Driven Development*), SCRUM i eXtreme programming (XP) – przedstawiciele *agile* (lekkich, sprawnych) procesów oraz USDP (*Unified Software Development Process*) czy wariant komercyjny USDP o nazwie RUP (*Rational Unified Process*) z lat 90. proponowany przez Rational IBM [Rekuć, 2008].

Różnice w ujmowaniu zagadnienia cyklu wytwórczego wynikają z odmiennego podejścia do problemu oraz odmiennej realizacji poszczególnych etapów projektowych. Zawsze ważne w procesie wytwórczym jest wybranie właściwej ścieżki

TABELA 1. ZESTAWIENIE METODYK OBIEKTOWYCH ORAZ ICH CECH CHARAKTERYSTYCZNYCH W ETAPACH PROJEKTOWYCH

			OODA (Booch)	OMT (Rumbaugh)	Objectory (Jacobson)
	Faza strategiczna	Określenie wymagań	-	+/-	+
	Modelowanie (analiza)				
Dokumentacja		Projektowanie	+	-	-
		Implementacja	+	-	-
	Instalacja	Testowanie	brak		
		Konserwacja			

spośród wielu zróżnicowanych technologii. Z reguły rodzaj cyklu wytwórczego wybiera twórca w zależności od tego, jakie wykonuje zadanie. Najkorzystniej jest na podstawie znanych ścieżek technologicznych opracować własną, ale dostosowaną do konkretnego, realizowanego zagadnienia.

Oprogramowanie GIS jest opracowywane z wykorzystaniem różnych środowisk implementacji, takich jak: język programowania (zbiór zasad składni, instrukcji, dzięki którym powstaje kod źródłowy programu) oraz system baz danych (są to rozwinięte metody i narzędzia organizacji, przechowywania oraz obróbki zbiorów danych). Należy rozstrzygnąć, jakie zastosować rozwiązania, mając na uwadze wymogi projektu architektury. Osiągnięcie celu, czyli wytworzenie oprogramowania, jest realizowane w sposób zależny od podejścia indywidualnego, czyli od właściwego doboru technologii tworzenia systemu. Podsumowując, w stworzeniu oprogramowania geodezyjnego ważnym aspektem jest cykl wytwórczy, który składa się z kilku etapów. Umożliwiają one pełne zaprojektowanie, a następnie użytkowanie programu. Każdy z tych etapów korzysta z własnej grupy metod, technik i narzędzi.

Ponadto systemy geodezyjne są systemami dedykowanymi, pracującymi w określonych środowiskach. Ważne jest zatem przestrzeganie zgodności z dokumentami funkcjonującymi w geodezji, takimi jak:

1. Normy ISO – na mocy prawa (zgodne z IEEE, wywodzą się z norm DIN).

2. Standardy, które zapewniają interoperacyjność (kompatybilność):

- IEEE (Institute of Electrical and Electronics Engineers – ogólnoświatowa organizacja wydająca kilkaset zaleceń rocznie);
- specyfikacje OpenGIS (OGC – Open Geospatial Consortium – utworzona w 1994 r. organizacja, której celem jest doprowadzenie do zintegrowania danych i funkcji w systemach informatycznych różnych producentów);
- OMG (Object Management Group: UML/XML);
- W3C (World Wide Web Consortium).

3. Dyrektywa INSPIRE (Infrastructure for Spatial Information in Europe).

4. Rozporządzenia, instrukcje, wytyczne techniczne, np. G-7, G-5.

PRZEGLĄD TECHNOLOGII, NARZĘDZI

W każdym z etapów projektowych stosuje się różne grupy narzędzi i techniki. Narzędzia te można podzielić na: systemy operacyjne (np. MS Windows, Linux, SUN), systemy bazodanowe (np. Oracle, MySQL), metodyki i narzędzia projektowania, języki programowania oraz zintegrowane środowiska programowe. Narzędzia CASE (*computer assisted/aided software/system engineering*) to komputerowo wspomagana inżynieria oprogramowania/systemów. Ze względu na zakres zastosowania stosuje się klasyfikację pakietów typu CASE na:

1. Pakiety narzędziowe (*Toolkits*) – do rozwiązywania problemów metod programowania, często w postaci dokumentów, analiz, wykresów, testów.

2. Pakiety zintegrowane (*Workbenches*) – do projektowania kompleksowego: narzędzia specyfikacji i interpretacji opisu systemu, generatory struktur baz danych, generatory programów wykonywalnych, programy dokonujące modyfikacji wersji systemu. Przykładowe funkcje narzędzi zintegrowanych ujmuje rys. 2 opracowany na podstawie ogólnego schematu narzędzi CASE [wg Jaszkiewicza, 1997].

Całkowity proces wytwarzania programu składa się z kilku etapów [Gratkowski, 2004; Jaszkiewicz,1997; Pirjanowicz, 2001; Szyjewski, 2005], z których każdy jest skutkiem poprzedniego – począwszy od opracowania koncepcji, a skończywszy na udoskonalaniu programu. Odpowiedni układ tych faz prowadzi do określonej kolejności działań zmierzających do utworzenia programu.

PRZEGLĄD METODYK

Pierwszym krokiem jest wybranie metody opracowania projektu. Ze względu na sposób strukturalizacji programu wyróżnić można dwie grupy metodyk, tj. strukturalne i obiektowe. Metodyka jest z kolei powiązana z notacją, która służy do dokumentowania poszczególnych etapów projektowych oraz jest niezbędnym środkiem komunikacji w grupie realizującej projekt [Olszak, Sroka, 2003]. Według Wesołowskiego [1999] metodyka jest zbiorem zasad i sposobów działania zmierzających do rozwiązania określonych problemów i osiągnięcia wyznaczonych celów.

1. Metodyki strukturalne łączą statyczny opis danych i procesów. Analiza strukturalna rozpoczyna się od budowy dwóch różnych modeli systemu: modelu danych oraz modelu funkcji. Te dwa modele są integrowane. Wynikiem jest model przepływu danych. Wśród metodyk strukturalnych najbardziej znane są: metodyka Yourdona (DeMarco i Ward/Mellor), J. Martina, SSADM (*Structured Systems Analysis and Design Method*), SADT (*Structured Analysis and Design Technique*).

2. Metodyki obiektowe są oparte na wyróżnianiu obiektów łącznie z operacjami. Podstawowym składnikiem jest diagram klas, który oprócz klas zawiera specyfikacje atrybutów i procedur, związki generalizacji, związki asocjacji i agregacji, liczności tych związków, różnorodne ograniczenia oraz inne oznaczenia. Jego uzupełnieniem są diagramy dynamiczne uwzględniające stany i przejścia pomiędzy nimi, diagramy interakcji ustalające zależności pomiędzy wywołaniami procedur, diagramy funkcjonalne (będące zwykle pewnym rodzajem diagramów przepływu danych). Metodyki obiektowe to: OMT (Rumbaugh, *Object Modeling Technique*), OODA (Booch, *Object-Oriented Analysis and Development*), OOSE (Jacobson, *Object-Oriented Software Engineering*). Połączenie tych trzech metodyk dało podstawy notacji UML (metodyka *Rational*).

Zestawienie metodyk i ich podejścia do poszczególnych faz zawiera tabela 1, gdzie „+" oznacza dobre podejście, natomiast „-" niewystarczające.

ŚRODOWISKA NARZĘDZIOWE WSPOMAGAJĄCE ETAP MODELOWANIA (ANALIZY)

Dobry projekt można uzyskać tylko wtedy, gdy metodyki, notacje i narzędzia będą wspomagały pracę projektantów. Same narzędzia nie narzucają metodyki,

określają one tylko notację. Zachowanie przyjętych standardów (np. w zakresie notacji, sprawdzenie poprawności i wzajemnej spójności modelu, przejrzystości, logicznego układu i spójności dokumentacji) to czynniki sukcesu fazy analizy.

Narzędzia do modelowania projektu można podzielić na programy do tworzenia diagramów oraz narzędzia CASE. Pierwsze z nich służą przede wszystkim do rysowania modeli. Ich zaletą jest prostota obsługi i niska cena, w związku z czym programy te nadają się do masowego użytku i szybkiego tworzenia modeli. Ogólnie narzędzia te mogą być wystarczające, jeżeli podstawowe znaczenie w trakcie prac ma graficzna reprezentacja procesów i ich dokumentacja. Wady omawianych programów ujawniają się natomiast przy próbach integracji modeli oraz integracji z innymi narzędziami. Zintegrowane środowiska służą zaś do definiowania założeń systemu informatycznego, na podstawie których generowana jest następnie struktura kodu źródłowego.

Duża liczba dostępnych narzędzi powoduje, iż wybranie odpowiedniego do tworzonego projektu wymaga od użytkownika szczegółowej analizy. Narzędzia te można podzielić ze względu na ich sposób licencjonowania. Przykładowe **open source** to: Ameos, AndroMDA, ArgoUML, BOUML, EclipseUML, ESS-Model, Fujaba, JGraph, Oracle JDeveloper, Umbrello UML Modeller, UMLGraph.

Natomiast do **komercyjnych** należą m.in.: Artiso Visual Case, CASE Studio, Code Logic, Component Architect, Database Architect, DTM Data Modeler, Enterprise Architect, Estimator/manager, FlowChart.NET, GoDiagram, Ideogramic UML, IntelliUML Teresa, iUML, J2U, MagicDraw, MDE for UM, MetaEdit+, MIA-Generation, ModelMaker, Object Domain, Objecteering, ObjectiF, Plastic, Poseidon, PowerDesigner, Prosa UML Modeler, Real-time Modeler, RFFlow Flowchart, Rose Developer, Simply Objects Modeler, SoftModeler, Together for Visual Studio.NET i Together Designer (Architect), UML Diagrammer, UML Studio, UML2COM, Visio Professional, Visual Studio.NET, Visual UML, Wilde Architecture Implementation, Win A&D, WithClass, UModel.

W tabeli 2 zostały przedstawione niektóre programy z drugiej grupy (wyboru dokonano według kryterium ich funkcjonalności). W związku z dużą liczbą zastosowanych w niej skrótów w następnym rozdziale zdefiniowano zagadnienia wymagające wyjaśnienia.

JĘZYKI ETAPU IMPLEMENTACJI

Język programowania to zbiór zasad składni i instrukcji, dzięki którym powstaje kod źródłowy programu. Procesor jest w stanie wykonywać program w kodzie maszynowym. Jednakże tworzenie programów w tym języku jest praktycznie niemożliwe. Dlatego programista używa języka zrozumiałego dla człowieka, a wyrażenia w tym języku są następnie kompilowane bądź interpretowane do postaci maszynowej.

Wzorce GoF (*Gang of Four*) – ich charakterystyczną cechą jest, że dotyczą konstrukcji programistycznych używanych do poprawy jakości tworzonego kodu. Wzorce te dotyczą współpracy obiektów w obrębie jednego systemu. Są podstawą piramidy, na której opiera się praca i rozwój programisty (http://jdn.pl).

Istnieje wiele rodzajów języków programowania. Można je podzielić na strukturalne i obiektowe. Odmiennym kryterium podziału jest zastosowanie języków (innych używa się do tworzenia programów multimedialnych, a innych do obliczeń numerycznych czy aplikacji sieciowych).

Obecnie programiści używają tzw. języków wysokiego poziomu trzeciej generacji. Są one bliższe składni języka naturalnego i nie są zależne od konkretnego procesora, przynajmniej na etapie samego programowania. Językami **strukturalnymi** wysokiego poziomu są na przykład:

- **FORTRAN** – skrót od *FORmulae TRANslator*, co oznacza: tłumacz wzorów. Język ten powstał w latach 50. w firmie IBM. Obecnie używane są rozbudowane i zestandaryzowane wersje tego języka.
- **Język C** – powstał do przeprogramowania UNIX-a, ale szybko zyskał popularność jako uniwersalny język programowania. Jego cechami są zwięzłość i elastyczność, przy jednoczesnym przerzucaniu dużej odpowiedzialności na programistę (nie ma np. wbudowanej kontroli indeksowania tablic).
- **PASCAL** – stworzony przez Niklausa Wirtha w 1971 r. Jeden z pierwszych języków strukturalnych, ma obecnie niewielkie zastosowanie w praktyce, ale jest wykorzystywany do nauki programowania.

Drugą grupę języków trzeciej generacji stanowią języki **obiektowe**:

- **C++** – rozszerzenie języka C, znane początkowo pod nazwą „C with Classes". Stworzył go już w 1983 r. Bjarne Stroustrup

TABELA 2. ZESTAWIENIE WYBRANYCH NARZĘDZI WSPOMAGAJĄCYCH ETAP MODELOWANIA

Narzędzie	Microsoft Visio + SP2	Altova UModel 2009	Enterprise Architect	MagicDraw UML	Case Studio 2
Producent	Microsoft	Altova	Sparx Systems	No Magic	Charonware
Wersja testowa	nie	30 dni	30 dni	shareware	30 dni
Odczyt/zapis modelu	VSD, rysunek oraz wzornik XML, zapisywanie również rysunku w DWG lub DXF	posiada specjalny typ diagramu do generowania XML Schema	projekt EAP, zapis XML Schema, technologia MDA	przywracanie Database Structure (Schema) via JDBC	Oracle 9i, 8i, IBM DB2 v.7.1, MySQL 4, 3.23, PostgreSQL v.7.2, MS Access 2000 and 97, MS SQL 2000, Sybase ASE 12.5, Paradox, Interbase
Zapis bazy	eksport modeli do plików zgodnych z XML wraz ze standardem XMI	XMI import/ eksport	generowanie skryptów DDL do automatycznego tworzenia struktury bazy danych	generowanie skryptów DDL do automatycznego tworzenia struktury bazy danych	Automatycznie generuje skrypt SQL, synchronizuje model z bazą Oracle oraz MSSQL
Generowanie struktury kodu	brak	Java, C#, Visual Basic .NET	transformacja do DDL, Java, C#, EJB, XSD	GoF, Java, C#, C ++, EJB, WSDL, XML Schema, CORBA IDL	
Integracja z innymi programami	posiada edytor VB; programy Microsoft Office; umożliwia obsługę innych formatów, np. PDF, XPS	Visual Studio, Eclipse	Visual Studio .NET, Eclipse	Eclipse, IBM's WebSphere Application Developer and Rational Application Developer, Borland's JBuilder, IntelliJ IDEA, NetBeans, Sun's Java Studio, BEA WebLogic Workshop	diagramy ER
Generowanie dokumentacji	definicja raportu VRD	RTF	RTF, dodatkowy generator raportów do HTML	HTML, PDF, RTF	HTML lub RTF

z Bell Laboratories, aby umożliwić programowanie zorientowane obiektowo.

● **Standard ANSI C++** – zatwierdzony w roku 1994 język hybrydowy, pochodna języka C. Łączy własności C niskiego poziomu, takie jak arytmetyka wskaźników, z konstrukcjami wysokiego poziomu, takimi jak: klasy, podklasy, hermetyzacja, funkcje wirtualne. Zastosowania na skalę przemysłową. Krytykowany z powodu wolnego tworzenia aplikacji, zawodności, słabej przenaszalności, dużego ryzyka wadliwego działania programów.

● **R++** – rozszerzenie języka C++ o nową konstrukcję – *rule* (stąd R w nazwie). *Rule* oznacza warunek i działanie wykonywane w przypadku, gdy ten warunek jest spełniony. Obecnie R++ jest zaimplementowany jako preprocesor.

● **Java** – język autorstwa J. Goslinga (firma Sun Microsystems), używany do oprogramowywania specjalizowanych mikroprocesorów, wzbogacania prezentacji danych zawartych w dokumentach HTML oraz do opracowywania samodzielnych aplikacji wielowątkowych i rozproszonych. Jego istotną własnością jest to, że programy kompiluje się nie do poziomu kodu maszynowego, ale do poziomu znakowego języka pośredniego, czyli tzw. apletów (*applets*), które są następnie interpretowane. Daje to efekt dużej przenaszalności programów oraz zwiększenia bezpieczeństwa (*security*), co jest szczególnie istotne w środowiskach rozproszonych, takich jak np. internet. Na języku Java oparty jest język J++ firmy Microsoft.

● **Visual Basic** – opracowany przez Microsoft, oparty na Basicu język programowania, a także graficzne środowisko programistyczne (jedno z najwcześniej udostępnionych – ok. 1990 r.). W językach typu *visual* programowanie w znacznym stopniu polega na umiejętnym operowaniu gotowymi elementami (obiektami), takimi jak przyciski i pola dialogowe, a następnie przypisywaniu tym obiektom pożądanego przez programistę wyglądu i sposobu zachowania się, czyli reakcji na dokonywane na tych obiektach operacje. W żargonie mówi się o reakcji obiektu na zdarzenie. Zdarzeniem jest np. kliknięcie myszą, a reakcją obiektu (np. okna) może być jego zamknięcie albo zminimalizowanie do ikony.

● **C Sharp** – język stworzony przez firmę Microsoft dla platformy .NET (architektem jest duński programista Anders Hejlsberg). Bardzo podobny do języka Java (wg źródeł: 70% Java, 10% C++, 5% Visual Basic, 15% nowych cech).

● **Object Pascal/Delphi**. Turbo Pascal – jedna z popularniejszych implementacji kompilatorów języka Pascal. Wersja 1.0 środowiska Turbo Pascal została wprowadzona na rynek w 1983 roku. Wersja 4.0 ukazała się w 1987 r. i wprowadziła pojęcie modułu i kompilacji warunkowej. W roku 1988 pojawiła się wersja 5.0 (wśród nowości znalazły się typy proceduralne i funkcyjne oraz możliwość debugowania kodu źródłowego). Wersja 5.5 wprowadziła nowy, rewolucyjny w odniesieniu do języka Pascal obiektowy typ danych. Ostatnia wersja oznaczona została numerem 7.0, a jej następcą stał się w 1995 r. Borland Delphi 1.0 wraz z językiem Object Pascal.

● **Inne języki** – Smalltalk, Python, Ada, Eiffel, OO-COBOL, Beta, CLOS, Modula-3, Objective-C, Trellis\Owl, Agora, Self, Sather, Actor, Sina, Dylan, DSM, Hetta, LENS.

TABELA 3. ZESTAWIENIE ZINTEGROWANYCH NARZĘDZI PROGRAMOWYCH

Język programowania	Licencjonowane środowiska			Darmowe środowiska
	Borland	Microsoft	Sun	
Delphi	KYLIX 2 (Linux) DELPHI	-	-	Dev-Pascal 1.9.2
Java	JBuilder6, STUDIO FOR JAVA	Visual Studio.NET	● Forte for Java ● Java Studio ● Java Dynamic Management Kit (JDMK)	● NetBeans - Java ● VIDE ● Gel RC20 ● jGRASP 1.5.3 ● Doxygen ● ECLIPSE
C/C++	C++BuilderTM 6	Visual C++ 6	-	● NetBeans - Java ● RHIDE (Robert Hoehne IDE dla DOS/ Windows/ Linux) ● Dev-C++ ● kDevelop 2.1.5 ● VIDE ● jGRASP 1.5.3 ● Doxygen ● ECLIPSE
C Sharp		Visual Studio.NET		
VB.NET/VB	-	Visual Studio 6	-	
J++	-	Visual J++ 6	-	

● NARZĘDZIA WSPOMAGAJĄCE ETAP IMPLEMENTACJI

Zintegrowane środowiska projektowe typu IDE (*Intergrated Development Environment*) uwzględniają większość potrzeb i postulatów programistów. Korzystanie ze standardowych, prostych narzędzi, np. notatnika, sprawia, że zamiast koncentrować się na rozwijaniu tego, co jest istotą w trakcie tworzenia oprogramowania, skupiamy się na tym, czy na pewno użyliśmy np. średnika czy wielkiej litery. Wiele środowisk projektowych IDE nie tylko identyfikuje potencjalne błędy i o nich informuje, ale również sugeruje sposób ich poprawienia. Automatyzuje tworzenie większości podstawowych zadań w poszczególnych etapach projektowania. W tabeli 3 przedstawiono zestawienie wybranych środowisk programowych (podziału dokonano według języka programowania oraz firm i najczęściej stosowanych, dostępnych pakietów).

● GEODEZJA Z INFORMATYKĄ

Budowa dowolnego oprogramowania dedykowanego, np. geodezyjnego, wiąże się z koniecznością zapoznania się z technologią inżynierii oprogramowania. Oznacza to, że aby stworzyć rozwiązanie postawionego problemu (np. transferu danych sieci uzbrojenia terenu), należy określić rozmiar przedsięwzięcia oraz zapoznać się z szerokim zakresem wiadomości na temat technologii projektowania systemów informatycznych. Połączenie wiedzy geodezyjnej i informatycznej daje rezultaty w postaci tworzonych rozwiązań programowych.

BEATA ŻERO
(Uniwersytet Warmińsko-Mazurski w Olsztynie)

Literatura

•Gaździcki J., 1990: Systemy Informacji Przestrzennej, PPWK Warszawa-Wrocław;
•Gratkowski T., 2004: Cykl życia projektu informatycznego, Konferencja Naukowa KNWS '04 „Informatyka – sztuka czy rzemiosło", Czocha;
•Jaszkiewicz A., 1997: Inżynieria oprogramowania, Helion, Gliwice;
•Kolbusz E., Olejniczak W., Szyjewski Z., 2005: Inżynieria systemów informatycznych w e-gospodarce, PWE Warszawa;
•Olszak C., Sroka H., 2003: Informatyka w zarządzaniu, Akademia Ekonomiczna w Katowicach;
•Rekuć L., 2008: Wytwarzanie oprogramowania;
•Subieta K., 1999: Wprowadzenie do obiektowych metodyk projektowania i notacji UML, Polsko-Japońska Wyższa Szkoła Technik Komputerowych, XI Górska Szkoła PTI Szczyrk;
•Wesołowski W.J., 1999: Inżynieria zarządzania, Wyższa Szkoła Handlu i Prawa, Warszawa.