Polski Przegląd Kartograficzny
Tom 37, 2005, nr 1, s. 13–22

WOJCIECH POMIANOWSKI
Instytut Geografii i Przestrzennego Zagospodarowania PAN
Warszawa

Semantyczna analiza oprogramowania GIS

Zarys treści. Brakuje metody opisu programów GIS, która pozwoliłaby poznać nie tylko ich funkcje i sposób komunikacji z użytkownikiem, ale również ich głęboką strukturę. Celem analizy semantycznej jest odkrycie tej struktury i zbadanie jej związku ze strukturą poznawczą dyscypliny macierzystej – geografii. Podstawową jednostką głębokiej struktury jest metafora.

Słowa kluczowe: teoria GIS, geoinformatyka, inżynieria oprogramowania, semiologia, nauki kognitywne, metafora

1. Skąd czerpiemy wiedzę o GIS?

Mimo istnienia od wielu lat rynku oprogramowania GIS oraz dużej grupy producentów, użytkowników i ekspertów, trudno jest znaleźć opracowania, które w rzetelny i wnikliwy sposób dokonują tak prostej rzeczy, jak opis tych systemów[1], nie wspominając już o ich porównaniach. Do tego rodzaju prac przykładane są najwyraźniej niższe kryteria jakościowe niż do prac naukowych, dlatego trzeba je zaliczać do kategorii publicystyki – mniej lub bardziej ciekawej, ale zawsze pozbawionej dyscypliny metodologicznej. Autorzy, których ambicją jest szczegółowe przedstawienie takich systemów, nie dysponując odpowiednimi narzędziami, poprzestają na zestawieniach właściwości (ang. *factsheet*), które zrobione niestarannie i mechanicznie, zamieniają się w bezużyteczne „wyliczanki". Jedynymi odbiorcami, którzy mogą z nich skorzystać są ci, którzy szukają konkretnej właściwości lub funkcji albo też chcą dokonać pod jej kątem porównania. Taka forma opisu nadaje się być może dla prostych produktów konsumenckich, ale na pewno nie dla systemów informatycznych o dużej złożoności. O ile można dość łatwo zdefiniować kilka istotnych i łatwych do zmierzenia cech proszku do prania albo telewizora, to nie da się tego zrobić w przypadku oprogramowania komputerowego. Nawet gdyby było to możliwe, to nie powstałby w ten sposób wiarygodny obraz systemu, obraz który oddawałby jego filozofię i pozwoliłby dokonać rzetelnej oceny. Oczywiście, w jeszcze mniejszym stopniu należy tego oczekiwać od firmowych materiałów informacyjnych lub reklamowych.

Największym pod względem liczebności źródłem wiedzy na temat systemów informacji geograficznej są prace poświęcone ich konkretnym zastosowaniom. Dominują one w literaturze i materiałach konferencyjnych w kraju i za granicą. Nie pomniejszając znaczenia aplikacyjnej strony GIS należy uznać, że opracowania te w niewielkim stopniu i tylko pośrednio pogłębiają naszą wiedzę o poszczególnych systemach i o całej dyscyplinie. Oprogramowanie jest w nich traktowane czysto instrumentalnie, co utrwala niesłuszny pogląd, że GIS jest tylko narzędziem. Przy okazji można zauważyć, że przewaga prac aplikacyjnych nad teoretycznymi jest w naszym kraju większa niż gdzie indziej, co powoduje, że Polska nie odgrywa na arenie światowej prawie żadnej roli w budowaniu nowej dyscypliny nauki, jaką jest geoinformatyka.

Kolejnym źródłem informacji są podręczniki do systemów. Współczesna maniera (która powstała pod trudnym do powstrzymania wpływem firmy Microsoft) nakazuje, aby zawierały one opis czynności prowadzących do wykonania pewnych „zadań" (ang. *tasks*) i to koniecznie metodą „krok po kroku". Są one napisane prostym, pogodnym językiem i wprowadzają tylko tyle uogólnień, ile jest absolutnie niezbędne do wykonania zestawu ćwiczeń. Jednak usuwając wszelkie pojęcia abstrakcyjne, autorzy powodują, że ich prace stają się mało wartościowe pod względem

[1] W artykule mowa o oprogramowaniu GIS uniwersalnego przeznaczenia, a nie o gotowych systemach wraz z danymi lub aplikacjach „pod klucz".

poznawczym. Mimo podobnej nazwy „podręcznik akademicki" i „podręcznik do systemu", pierwszy zawiera syntezę wiedzy, podczas gdy drugi przypomina instrukcję obsługi urządzenia mechanicznego.

Inną płaszczyzną, na której odbywa się zbieranie informacji i ocena systemów, jest bezpośrednia rozmowa. Użytkownicy mówią o oprogramowaniu skupiając się na funkcjach, które ono realizuje, sposobie realizowania pewnych czynności lub stosowania narzędzi. Dokonują więc opisu na „płytkim" poziomie, bez wnikania w system pojęciowy, który w istocie steruje oprogramowaniem. Kiedy jednak tematem rozmowy jest ogólna użyteczność systemu w kontekście dużego projektu lub długotrwałego użytkowania, styl i język ulegają zmianie – pojawia się próba analizy na głębszym poziomie. Wyraźna różnica w percepcji oprogramowania wynika nie tylko z doświadczenia – są użytkownicy z wieloletnim doświadczeniem, których okoliczności nigdy nie zmusiły do dogłębnego zrozumienia swojego systemu. Decydującym czynnikiem jest odpowiedzialność. Osoby odpowiedzialne za duże projekty starają się dotrzeć do głębokiej struktury, ponieważ od jej poznania może zależeć ich sukces lub porażka zawodowa.

Istnieją dwa momenty, w których następuje ostateczna weryfikacja wiedzy i zrozumienia potencjalnego użytkownika na temat danego systemu oprogramowania. Pierwszym jest decyzja o zakupie, drugim – podjęcie samodzielnej pracy w systemie. W obu przypadkach okazuje się, że źródła pisane nie wystarczają. Użytkownicy chcą zobaczyć oprogramowanie w rzeczywistych warunkach pracy, chcą je „poczuć" i samodzielnie wypróbować. Jest to chyba najlepszy dowód na to, że panujące obecnie sposoby opisu nie dostarczają wystarczającej wiedzy na temat oprogramowania.

2. Czy istnieją dobre narzędzia opisu?

Trudności w znalezieniu właściwego języka i metody opisu oprogramowania nie są wyjątkowe w dziedzinie GIS. Dotyczą właściwie wszystkich programów – z tym, że zaczynają stanowić poważny problem dopiero w przypadku specjalistycznych systemów o wysokim stopniu złożoności. Czy informatyka dostarcza narzędzi do tego rodzaju analizy? Pierwszym kandydatem mogłaby być inżynieria programowania (ang. *software engineering*) – dyscyplina z dużymi tradycjami, która wypracowała wiele formalnych metod specyfikacji oprogramowania, wyrażonych za pomocą specjalnych języków lub schematów. Mogą one opisywać system pod różnym kątem, np. stanów, relacji między składnikami, ciągów czynności (ang. *workflow*) (M. R. Little, L. Nigay 2001). Narzędzia inżynierii służą jednak przede wszystkim do optymalizowania procesu programowania i zagwarantowania formalnej poprawności kodu, nie zaś – do analizowania struktur poznawczych użytkownika i tworzenia odpowiadających im struktur w oprogramowaniu. Bliższą tego celu może być technika nazywana *domain modelling* (modelowanie dziedziny), która funkcjonuje na pograniczu informatyki i nauk o organizacji. Dzięki niej można przeanalizować strukturę organizacji, znaleźć dla niej zbliżoną strukturę systemu informatycznego i przedstawić ją w standardowej notacji, na przykład UML – Universal Modelling Language (G. Booch i inni 1999). Diagram albo inny formalny zapis, będący rezultatem takiej analizy, jest precyzyjny i jednoznaczny do tego stopnia, że nadaje się do automatycznego przetwarzania, ale jest wyrwany z kontekstu dyscypliny i słabo czytelny dla osoby nie znającej jego konwencji. Drugi problem wynika z faktu, że modelowanie dziedzin stosuje się do konkretnych przypadków układu „organizacja – system informatyczny", a nie do bardziej uniwersalnego układu „dziedzina wiedzy – oprogramowanie". Mimo, że prawdopodobnie udałoby się przedstawienie struktury pojęciowej konsumenckiego oprogramowania GIS przy użyciu UML, to jednak trudno byłoby uzyskać akceptację dla takiej metody w środowisku geografów.

3. Płytki i głęboki poziom opisu

Informatyka koncentruje się na oprogramowaniu, a wiedzę o użytkowniku musi czerpać z zewnątrz. Przyczyna takiego stanu rzeczy jest dość oczywista: „twarde" dyscypliny informatyki rozwijają się od 50 lat, „miękkie" – od 30, czyli od czasu upowszechnienia komputerów osobistych. Najważniejszy system klasyfikacyjny – ACM (Associacion for Computing Machinery) – nie wymienia żadnej odrębnej poddyscypliny zajmującej się tym zagadnieniem z wyjątkiem tzw. komunikacji z użytkownikiem (ang. *user interface*). Przedmiotem jej zainteresowania jest interakcja użytkownika z oprogramowaniem za pośrednictwem elementów ekranowych, myszki i klawiatury. Odkryte prawidłowości odwołują się do prostych odruchów i uniwersalnych zasad percepcji barw, kształtów i są ważne dla każdego rodzaju oprogramowania, zarówno przeznaczonego do prac biurowych jak i do komponowania muzyki lub redagowania map.

Dlatego powinno się ją określać mianem płytkiej warstwy komunikacji. Standaryzacja na tym polu leży w interesie dużych dostawców oprogramowania systemowego, dlatego publikują oni podręczniki, które upowszechniają dobre praktyki programistyczne (np. Apple 2002, Microsoft 2001).

W ostatnim czasie podejmowane są próby włączenia osiągnięć psychologii poznawczej (ang. *cognitive psychology*) i semiologii do procesu analizowania i projektowania oprogramowania (P.A. Chalmers 2003). Badania te kierują się w stronę głębokich struktur, które były już wcześniej odkryte przez Jeana Piageta i Jerome Brunera w procesie uczenia się, procesów kodowania-odkodowywania i semiozy (C.L. Nehaniv 2000). Jeśli bowiem struktury te są obecne w społecznym otoczeniu człowieka, to niewątpliwie można ich szukać również w otoczeniu informatycznym. Wyznaczając w 1968 roku program badawczy semiologii w *Nieobecnej Strukturze*, Umberto Eco wymienił 14 dziedzin jej zastosowania (semantyk), od komunikacji wzrokowej do kodów muzycznych. Nie wymienił jednak inżynierii oprogramowania. W ciągu następnych lat semiologia znalazła szerokie zastosowanie w badaniu różnych form komunikacji społecznej, a następnie została na trwałe zaadaptowana przez krytyków literatury, filmu, architektury i innych rodzajów sztuki. Jest więc ze wszech miar uzasadnione, aby wyznaczyć kolejną taką dziedzinę – semantykę oprogramowania, a w jej ramach – szereg specjalistycznych semantyk opisujących systemy w różnych dyscyplinach. Trzeba podkreślić, że o ile informatyka, psychologia poznawcza i semiologia mogą dostarczać narzędzi badawczych, to jednak ciężar analizy *swojego* oprogramowania pod kątem *swoich* struktur pojęciowych powinni podjąć sami przedstawiciele specjalistycznych dziedzin, ponieważ nikt nie zna się na nich tak dobrze, jak oni sami. Tymczasem geografowie i kartografowie (zwłaszcza o tradycyjnym wykształceniu, szczególnie w Polsce) unikają tego tematu uznając najwyraźniej, że jest on zbyt techniczny. Nie zadają fundamentalnych pytań, takich jak: czy schemat pojęciowy oprogramowania GIS jest odpowiedni? czy znajdują w nim znajome koncepcje, czy może również nowe – warte przyjęcia lub odrzucenia? Więcej jest uwag na temat szczegółów realizacji pewnych metod lub użyteczności GIS jako narzędzia do opracowywania starych problemów. Niestety, takie podejście prowadzi tylko do biernego konsumowania technologii GIS, a nie do jej aktywnego kształtowania.

Granica między płytką i głęboką warstwą nie jest jasno określona. Jest znamienne, że prace, które rozpoczynają się od rozważań nad manipulacją obiektami GIS za pomocą myszki (np. J.R. Richards, M. Egenhofer 1995), prowadzą niemal zawsze do podstawowych pytań o naturę mapy i informacji geograficznej. Również granice tematyczne są płynne: część z zagadnień, do których mogliby rościć sobie prawo kartografowie, jak na przykład „nawigowanie" i wyszukiwanie na mapie, jest badana przez psychologów w oderwaniu od kartografii (np. M.A. Walker i inni 2004). Wraz z upowszechnieniem map w oprogramowaniu i urządzeniach powszechnego użytku, tendencja ta będzie się z pewnością nasilać. Istnieją również przykłady przeciwnego podejścia (D. R. Montello i inni 2003), w którym modeluje się komunikację z użytkownikiem (płytką warstwę), wykorzystując koncepcję bliskości geograficznej i metody kartograficzne.

4. Znaczenie głębokich struktur

Mówiąc o głębokich strukturach, porzucamy te dziedziny psychologii, które traktują człowieka jako element układu bodziec – reakcja. Przyjmujemy założenie, że użytkownik jest świadomą istotą wyposażoną w wiedzę i doświadczenie zawodowe w jakiejś specjalizacji, składające się razem na strukturę poznawczą. Pod pojęciem struktury poznawczej należy rozumieć coś innego, niż szereg encyklopedycznych definicji lub reguł, a mianowicie – skomplikowaną siatkę pojęć, połączoną wzajemnymi relacjami i uzupełnioną o katalog „przypadków użycia" i wyjątków. Użytkownicy tego rodzaju, nazywani (według M.F. Costabile i innych 2003) „d-ekspertami" (od ang. *domain expert*), stanowią grupę stawiającą oprogramowaniu szczególne wymagania. D-eksperci nie realizują „zadań" o charakterze utylitarnym, prowadzących do wykonania ściśle określonego celu, tak jak to czyni personel biurowy; ich celem jest raczej szeroko pojęte modelowanie rzeczywistości.

W przypadku GIS, wzorcowym d-ekspertem jest geograf. Użytkownicy wywodzący się z innych dziedzin „wcielają się w skórę" geografa i starają się z lepszym lub gorszym skutkiem naśladować jego metodę pracy. Strukturę poznawczą geografa wyznaczają pojęcia kartograficzne takie jak mapa, legenda, zmienna, metoda prezentacji kartograficznej oraz pojęcia bardziej specyficzne dla jednej z poddziedzin geografii, np. zlewnia, płat, stok, potencjał, region. Struktury te stanowią same w sobie przedmiot intensywnych badań geoinformatyki od co naj-

mniej 10 lat. W tym miejscu skupimy się jednak na nieco innym zagadnieniu – sposobie, w jaki trafiają one do oprogramowania GIS.

Nie ulega wątpliwości, że projektanci oprogramowania muszą nawiązać do struktury poznawczej macierzystej dyscypliny – stworzyć w oprogramowaniu jej odpowiednik. Pierwszym krokiem jest przełożenie struktury pojęciowej eksperta na język zrozumiały dla laika (w tym – projektanta oprogramowania). Badania M. Rauterberga i M. Hofa (1995) dowodzą, że odbywa się to za pośrednictwem metafor. Jest zupełnie naturalne, że istnieje tendencja do wykorzystywania tej właśnie gotowej metaforycznej struktury w oprogramowaniu, o czym będzie mowa dalej.

Kiedy rozpatrujemy dwie równoległe struktury, najważniejsze pytanie brzmi: „czy odwzorowanie między nimi jest i czy powinno być wierne?" Należy postulować, aby głęboka struktura oprogramowania GIS była w jak największym stopniu zintegrowana ze strukturą pojęciową geografii, ponieważ ułatwia to pracę specjalistom, eliminuje lukę pokoleniową i przyczynia się do zachowania dorobku wiedzy. Nie można jednak oczekiwać stuprocentowej zgodności.

Po pierwsze, aby dobrze zdefiniować odwzorowanie, należy najpierw mieć dobrze zdefiniowaną strukturę wyjściową. Mimo, że podejmowane są próby modelowania struktury pojęciowej geografii, np. przy pomocy schematów UML (np. B. Brodaric, M. Gahegan 2002) lub formalnych języków (np. W. Kuhn 2002), to jest niemal pewne, że dyscyplina tak heterogeniczna i nieuporządkowana jak geografia nie wypracuje nigdy kompletnego i spójnego schematu pojęciowego, który byłby następnie przedmiotem poprawnie przeprowadzonej operacji odwzorowania. Można wręcz zaobserwować odwrotne zjawisko – struktura docelowa zrealizowana w GIS jest często bardziej precyzyjna niż struktura wyjściowa. Jest to zupełnie zrozumiałe: potrzeba systematyzacji i abstrahowania jest wymuszona przez ścisły i nieuznający wyjątków proces projektowania GIS. Daje to okazję do zdefiniowania nowych pojęć, zmiany znaczenia starych terminów i przebudowy całej struktury poznawczej – z tego właśnie powodu obserwujemy zwrotny wpływ GIS na kartografię i geografię.

Po drugie, zagwarantowanie całkowitej wierności byłoby niecelowe, ponieważ samo zastosowanie komputera daje możliwość poszerzenia modelu świata użytkownika o nowe narzędzia i możliwości. Przykładem może być poszerzenie repertuaru metod prezentacji kartograficznej, które doprowadziło do tego, że ich klasyfikacja w starej, podręcznikowej postaci jest już nie do utrzymania. Innym przykładem może być zmiana nastawienia do reguł stosowania odwzorowań kartograficznych: od kiedy mapa przestała pełnić rolę pierwotnego źródła informacji o położeniu obiektów, problem zniekształceń przestał budzić żywe zainteresowanie kartografów.

Kolejną kwestią wartą rozważenia jest stopień ogólności struktury poznawczej, która powstaje w oprogramowaniu. Można tu wyróżnić przynajmniej trzy klasy systemów:

1) aplikacje, które tworzy się dla rozwiązania konkretnego problemu (np. model zlewni Zgłowiączki);

2) programy obsługujące całą klasę problemów (np. modelowanie zlewni);

3) uniwersalne programy GIS (np. ArcGIS), uniwersalne standardy (np. OpenGIS).

Systemy obejmujące wąski wycinek struktury pojęciowej macierzystej dyscypliny (1) są łatwiejsze do zaprojektowania, niż systemy ogólne (3). Mogą być one wprawdzie trudne do zrealizowania z powodu wyrafinowanych algorytmów, ale ich głęboka struktura jest prostsza. Systemy uniwersalne muszą spełniać wymagania szerszego grona użytkowników z nakładającymi się, ale nie identycznymi strukturami pojęciowymi. W trakcie ich projektowania (albo dalszego rozwoju) niezbędny jest więc proces abstrahowania, który polega na odnajdywaniu związków i podobieństw między pojęciami, a następnie tworzeniu nowych, bardziej ogólnych pojęć; np. podobieństwo między ruchem wody w obrębie zlewni a ruchami migracyjnymi może prowadzić do powstania uogólnionego pojęcia przepływu. Nowe pojęcie może przypaść do gustu innym użytkownikom (np. zoogeografom). Obecność pojęć abstrakcyjnych w oprogramowaniu podnosi jego jakość i zakres zastosowań, ale poza walorami praktycznymi ważniejsze są racje fundamentalne. Abstrahowanie w czasie projektowania systemu nie różni się niczym od abstrahowania w nauce, w której jego konieczność jest pewnikiem nie wymagającym udowodnienia. Jest z nim związany kolejny postulat, znany w nauce pod nazwą „brzytwy Ockhama" – redukowanie schematu pojęciowego do niezbędnego minimum, a więc takiego, które wystarcza do wyjaśnienia problemu. Podobnie bez uzasadnienia przyjmiemy postulat, by głęboka struktura oprogramowania była, tak jak teoria naukowa, logiczna, spójna i pozbawiona niekonsekwencji. Nauka poszukuje zwięzłości, ogólności i elegancji po prostu dlatego, że uznaje je za dobre; podobnie powinni czynić programiści.

5. Problemy z głęboką strukturą

Niedopasowanie struktury poznawczej użytkownika i oprogramowania jest źródłem poważnych problemów. Przyczyną może być jeden lub kilka z poniższych czynników jednocześnie:

1) źle zaprojektowana struktura oprogramowania,

2) źle napisana dokumentacja,

3) błędna lub niekompletna struktura poznawcza użytkownika,

4) niechęć użytkownika do zrozumienia głębokiej struktury programu.

Warto zauważyć, że współczesna tendencja do maksymalnego upraszczania dokumentacji daje fatalne skutki. Następuje dość szybki przyrost praktycznych umiejętności użytkownika w początkowej fazie poznawania systemu, co powoduje powstanie wrażenia dobrej znajomości całego oprogramowania – wrażenia, które jest często całkowicie nieuzasadnione. Nie następuje bowiem dogłębne zrozumienie kluczowych pojęć i nabycie głębszej wiedzy.

Niezależnie od przyczyny, użytkownik, który nie rozumie oprogramowania, jest narażony nie tylko na uciążliwość lub wolne tempo pracy (jak to ma miejsce w przypadku źle zaprojektowanego systemu menu), ale na znacznie poważniejsze konsekwencje: uszkodzenia danych, zmarnowany wielki wkład pracy, niemożność osiągnięcia stawianego sobie celu, czyli coś, co łącznie można nazwać syndromem poznawczego załamania. Syndrom ten rzadko występuje w krańcowej postaci, prowadzącej do całkowitej rezygnacji z pracy w jakimś systemie. Najczęściej przyjmuje formę przewlekłą, ze stanami nasilenia i stanami relaksacji. Jest wręcz zdumiewające, jak wielka jest grupa użytkowników, którzy nauczyli się „stąpać po kruchym lodzie", a więc omijać niezrozumiałe dla siebie pojęcia i radzić sobie z problemami. Sposób postępowania takich użytkowników jest bardzo charakterystyczny. Po pierwsze, wypracowują oni sekwencje typowych czynności, którym odpowiadają ścieżki poruszania się po programie i pilnie się ich trzymają (niekiedy do tego stopnia, że za każdym razem rozpoczynają ścieżkę od otworzenia całego programu). Po drugie, posługują się metodą prób i błędów, porównując otrzymany rezultat ze swoimi oczekiwaniami albo podobnymi rezultatami otrzymanymi w przeszłości. Ten sposób może być skuteczny tylko w przypadku prostych zadań i niewielkich zbiorów danych, ale traci całkowicie skuteczność, kiedy dane są skomplikowane, trudne do wizualnej weryfikacji, a zbiory wielkie.

Syndrom poznawczego załamania objawia się wytworzeniem kilku mechanizmów obronnych:

1) „ja to robię inaczej": radzenie sobie drogą okrężną – skuteczną, ale wymagającą większego nakładu pracy;

2) „to można gdzieś ustawić, ale nie mam czasu": rezygnacja z dostosowania systemu do własnych potrzeb;

3) „na szczęście zrobiłem kopię": wytwarzanie nadmiernej liczby kopii zapasowych nie na wypadek awarii systemu, ale – własnej pomyłki;

4) „pokaż mi tylko, gdzie kliknąć": skupienie się wyłącznie na mechanicznych aspektach wykonywanych czynności, bez próby ich zrozumienia.

Psychologia kognitywna nie wypracowała jeszcze metod badania tego rodzaju frustracji. O ile testy, metody ilościowe i pojęcie „efektywności" (ang. *effectiveness*, *performance*), zdają dobrze egzamin przy badaniach płytkiej warstwy (A.A. Anderson 1996), to trudno je zastosować do badań głębokich struktur poznawczych na styku użytkownik – oprogramowanie.

Jedną z cech nowych tendencji w komunikacji z użytkownikiem jest daleko posunięta interaktywność, wykraczająca niekiedy poza rzeczywiste potrzeby. Interaktywność nie jest niczym innym, jak możliwością natychmiastowej weryfikacji swoich czynności. Jeśli zestawimy to z charakterystycznym objawem syndromu poznawczego załamania, dojdziemy do wniosku, że producenci oprogramowania, starając się ułatwić życie swoim klientom, sami wpędzają ich w problemy. Użytkownicy tracą motywację do poznawania systemu, ponieważ wolą na bieżąco sprawdzać rezultaty swoich poczynań. Zaczynają się zachowywać podobnie do prostych robotów, których dynamiką steruje z jednej strony silnik, z drugiej zaś sensor przeszkody. Taka strategia nie może oczywiście prowadzić do twórczego wykorzystania komputera. Z drugiej strony, właśnie brak możliwości weryfikacji wyników, może – paradoksalnie – zmusić użytkownika do sięgnięcia po bardziej zaawansowane i abstrakcyjne koncepcje. Tego rodzaju krytyczny punkt występuje na przykład w procesie wykrywania i korygowania rozbieżności między dwoma „równie wiarygodnymi" materiałami kartograficznymi w postaci cyfrowej. Jeśli nie ma trzeciego, jeszcze bardziej wiarygodnego materiału, który mógłby grać rolę arbitra, użytkownik–kartograf musi posiąść niezbędną wiedzę teoretyczną i zrozumieć kolejne etapy przekształceń, jakim uległy dane, aby poradzić sobie z problemem. Dopiero kiedy tak się stanie, może z przekonaniem zawierzyć jednemu albo drugiemu źródłu i nie musi odwoływać się do empirycznej weryfi-

kacji. Innym przykładem może być mechanizm aktualizowania mapy na podstawie informacji z bazy danych. Jeśli mapa składa się z kilkudziesięciu do kilkuset tysięcy elementów, to trudno sobie wyobrazić wizualną weryfikację[2]. Ten, kto w pełni zrozumiał, na czym polega procedura aktualizacji, nie musi sprawdzać wyników, ponieważ z góry wie, że są dobre. Ten, kto go nie zrozumiał, nie jest nawet w stanie stwierdzić, gdzie popełnił błąd.

To, czy syndrom poznawczego załamania przyniesie duże szkody, zależy od stopnia komplikacji oprogramowania. W przypadku przeglądarki internetowej efekt syndromu prawie nie występuje, a jego ewentualne skutki są mało istotne. Natomiast stopień komplikacji systemów przeznaczonych do pracy profesjonalnej jest znacznie wyższy, zatem prawdopodobieństwo wystąpienia syndromu rośnie. Zaawansowane programy GIS wymagają od użytkowników wiedzy dotyczącej kartografii i baz danych oraz umiejętności graficznych. Najbardziej szkodliwą metodą zachęcania klientów do sięgnięcia po nowe oprogramowanie jest przekonywanie ich, że system jest prosty w obsłudze. Samo sformułowanie, które sugeruje, że mamy do czynienia z urządzeniem podobnym do pralki lub telefonu komórkowego, jest głęboko nietrafne, ponieważ GIS jest oprogramowaniem do modelowania rzeczywistości, a nie narzędziem. Użytkownicy zabierający się do skomplikowanych problemów nie powinni oczekiwać prostych rozwiązań.

Na koniec spójrzmy na problemy tak, jak je widzi druga strona – producent oprogramowania. Każdy system ulega rozwojowi. W jego trakcie uwidacznia się różnica między płytką i głęboką warstwą oprogramowania: unowocześnienie stylistyki lub dodawanie nowych funkcji jest stosunkowo mało kosztowne, dlatego że zakres zmian jest pod kontrolą, a problemy są odseparowane. Producenci stosują więc często ten zabieg w celu odmłodzenia starzejących się systemów. Jednak w trakcie rozwoju następują również okazje do przemodelowania głębokiej struktury. Jedne z nich są wymuszane potrzebami użytkowników lub całego rynku, inne koniecznością naprawienia błędów lub niekonsekwencji. Producenci podchodzą z dużą ostrożnością do takich rewolucyjnych zmian (przykładem jest 8 linia ArcGIS). Przebudowa głębokiej struktury jest kosztowna i ryzykowna, ponieważ dotyka wszystkich składników systemu, może być ponadto źle odebrana przez użytkowników, ponieważ zmieniają się znaczenia pojęć i ich powiązania. Niestety, pozostanie przy starej strukturze uniemożliwia dalszy rozwój produktu albo powoduje narastanie „wieży Babel" – nieskoordynowanego zbioru różnych funkcji lub osobnych podsystemów bez łączącego je logicznego szkieletu. Kluczowym czynnikiem mającym wpływ na długowieczność systemu jest obecność abstrakcyjnych pojęć od samego początku jego istnienia.

6. Analiza głębokich struktur

Zaproponowaną w artykule metodę można nazwać analizą semantyczną oprogramowania. Polega ona na rozbiorze oprogramowania na czynniki pierwsze, a następnie badaniu ich znaczeń, właściwości, budowy, wzajemnych relacji oraz funkcjonowania w kontekście własnej dyscypliny oraz szerszym, ogólnokulturowym. Jej najbardziej charakterystyczną cechą jest odwrócenie kierunku patrzenia w stosunku do inżynierii programowania: stara się ona odszukać i wyodrębnić takie struktury, które widzi i rozumie (lub mógłby widzieć i rozumieć) użytkownik, natomiast funkcjonalność, poprawność i niezawodność programu traktuje drugorzędnie.

Istnieją dwa pola zastosowania metody semantycznej. Pierwszym jest proces projektowania nowego systemu, w którym na samym początku następuje świadome odnalezienie analogii i budowanie na ich podstawie składników systemu o jasno określonych znaczeniach[3]. Drugim jest analiza istniejącego systemu; wydaje się, że dobre byłoby w tym przypadku określenie dekonstrukcja oprogramowania. Metoda jest płodna poznawczo – wyodrębnienie prostych składników pozwala porównać je z pojęciami znanymi z życia codziennego albo z wiedzy specjalistycznej. Pozwala uzasadnić proste pytania: „co właściwie znaczy X w moim programie" (gdzie X oznacza pojęcie lub byt programistyczny) albo „dlaczego mogę z X zrobić to, a nie mogę zrobić tamtego". Pytania, które w innym kontekście mogłyby być odrzucone jako śmieszne, sięgają często od razu do sedna sprawy – jeśli tylko udaje się znaleźć na nie odpowiedzi, to mówią one więcej, niż statyczny opis. Po metodzie semantycznej można oczekiwać dwóch pozytywnych rezultatów: pogłębienia wiedzy użytkowników i zapoczątkowania oświeconej krytyki, podobnej do tej, jaką stosuje się w odniesieniu do zjawisk kulturowych lub estetycznych. Skoro

[2] W tym miejscu pomijamy poprawność wykonania operacji czyli wiarygodność systemu i koncentrujemy się tylko na tym, czy wyniki są zgodne z oczekiwaniami użytkownika.

[3] Autor próbował zastosować takie podejście projektując od nowa swój system informacji geograficznej AVISO.

bowiem spędzamy w wirtualnym świecie coraz więcej czasu, to trzeba zacząć go traktować jako immanentną część świata kultury i wnikliwie mu się przyglądać.

Analiza semantyczna nie dostarcza z pewnością kompletnego opisu systemu. Należy się spodziewać zarzutu, że pomija ona ważne funkcje lub pożyteczne udogodnienia, realizowane przez opisywany system. To prawda – analiza semantyczna jest tylko metodą komplementarną w stosunku do opisu funkcji i opisu komunikacji z użytkownikiem (*user interface*). Pełny opis można dopiero uzyskać stosując trzy spojrzenia jednocześnie. Warto jednak zauważyć, że o ile opis semantyczny radzi sobie względnie dobrze bez opisu funkcjonalnego, to opis funkcjonalny pozbawiony wyjaśnienia głębokich struktur jest równie „niesprawiedliwy", a ponadto bezużyteczny. Rozważmy następujący przykład: dokonujemy prostego porównania liczby odwzorowań, które realizują dwa programy GIS. Pojęcie odwzorowania ma jednak w obu systemach różne znaczenie: w pierwszym oznacza tylko typ przekształcenia matematycznego, w drugim oznacza przekształcenie wraz z zestawem na stałe przypisanych parametrów. Odwzorowań w drugim rozumieniu jest, z definicji, o rząd wielkości więcej niż w pierwszym, ponieważ ich liczba jest wynikiem iloczynu możliwych przekształceń i przyjętych przez producenta konkretnych wartości parametrów. Drugi system wyjdzie zwycięsko z tego porównania, mimo że posługuje się pojęciem o mniejszym stopniu ogólności, a jego faktyczna użyteczność jest mniejsza. Błąd w analizie polegał na tym, że porównywano rzeczy, których nie wolno porównywać. Bez wyjaśnienia logiki programu nie można w ogóle zacząć rozmowy o jego funkcjonalności.

7. Metafory w oprogramowaniu

Analiza semantyczna musi mieć swój początek w rozbiorze oprogramowania i wydzieleniu podstawowych jednostek niosących znaczenie użytkownikowi. Taką jednostkę najlepiej określa termin metafora (wprowadzony wcześniej w odniesieniu do procesu przekazywania wiedzy). Metafora jest umysłowym narzędziem, dzięki któremu człowiek prowadzi dialog z oprogramowaniem, podobnie jak jego ręce są mechanicznym narzędziem dialogu z klawiaturą. Metafora składa się z pojęcia oraz jego reprezentacji w programie. O ile samego pojęcia możemy używać w dyskursie, to do pracy potrzebujemy jego postaci operacyjnej – takiej, z którą można coś zrobić. Właśnie reprezentacja stoi za pojęciem i nadaje mu „materialność" w taki sposób, aby użytkownik mógł wykonywać przy jego pomocy lub na nim pewne operacje. Metafory można dostrzegać zarówno w płytkiej, jak i głębokiej warstwie oprogramowania (w odniesieniu do tych pierwszych stosuje się w literaturze angielskojęzycznej trudne do przetłumaczenia terminy *widget* lub *applet*). W dalszej części tekstu ograniczymy się jednak tylko do tej drugiej, bowiem pierwsza doczekała się już licznych opracowań.

Użycie terminu metafora w odniesieniu do oprogramowania jest w pewnym sensie koniecznością, wynikającą z braku lepszego terminu. W przedstawionym tu rozumieniu, metafora jest czymś innym niż figura stylistyczna, spotykana na co dzień w języku, a opisana w *Poetyce* Arystotelesa jako narzędzie retoryki. Obie łączy występowanie analogii między światem realnym a światem wirtualnym, jakim jest język lub oprogramowanie. Metafory językowe podlegają jednak mniejszym ograniczeniom. Ich budowanie w procesie konstruowania wypowiedzi następuje spontanicznie i nie podlega konwencjom w takim stopniu, w jakim dotyczy to metafor w oprogramowaniu. G. Lackoff i M. Johnson (1980) nadają metaforze inne, daleko szersze znaczenie. W ich wydaniu staje się ona strukturą poznawczą, wywodzącą się z otaczającej człowieka rzeczywistości i przeniesioną do języka – nie wywodzi się zatem z jakiejś szczególnej dyscypliny (zastosowanie rozróżnienia płytki – głęboki byłoby tu mylące). Podobnego mechanizmu przeniesienia można się doszukiwać w relacji świat realny – świat wirtualny.

Metafory dotyczą najczęściej rzeczy – pojęcie jest określone rzeczownikowo, a jego reprezentacja ma formę graficzną. Takie metafory będziemy nazywać rzeczowymi. Wizualna reprezentacja takiej metafory może przyjmować najrozmaitsze formy – od prostych, skonwencjalizowanych, np. ikona koperty, do maksymalnie realistycznych, np. trójwymiarowy model budynku. W przypadku bardziej skomplikowanych metafor stosuje się więcej niż jedną reprezentację, która może się zmieniać w zależności od kontekstu.

Rzadziej mamy do czynienia z metaforami czynnościowymi, w których pojęcie jest określone czasownikowo, a jego reprezentacja jest również czynnością. Przykładem jest otwieranie i zamykanie lub operacja przeciągnij–i–upuść (ang. *drag and drop*). Czynność jest zawsze wykonywana w odniesieniu do innej, rzeczowej metafory, np. dokumentu. Metafory czynnościowe są bardziej skonwencjonalizowane niż rzeczowe: umówiliśmy się, że dokument zamykamy przez kliknięcie w specjalny przycisk, chociaż

w materialnym świecie zamykamy rzeczy na sto różnych sposobów.

Metafora opiera się zawsze na analogii między bytem powołanym do życia w oprogramowaniu, a bytem istniejącym „na zewnątrz" – w świecie materialnym lub otoczeniu kulturowym. Siła i jednoznaczność tej analogii decydują o jakości metafory. Pewne metafory, takie jak list, skrzynka pocztowa, kartka papieru są natychmiast zrozumiałe, ponieważ odwołują się do obiektów materialnych i powszechnie znanych. Inne, takie jak plik lub folder, będą od razu zrozumiałe tylko dla pracownika biurowego, natomiast ich akceptacja przez innych użytkowników jest w dużej mierze kwestią konwencji i przyzwyczajenia (metafory te ugruntowały się przez wieloletnie zastosowanie w systemach operacyjnych i obecnie wydają się naturalne). Metafora może być specyficzna dla jednej dziedziny, albo mieć szersze zastosowanie. Istnieje na przykład grupa metafor funkcjonujących jednocześnie w GIS oraz w innych rodzajach oprogramowania, np. dokument, styl, rozkład (ang. *layout*), obiekt. Mogą one sprawiać trudność użytkownikom, ponieważ mimo tej samej nazwy i podobieństwa niektórych aspektów zachowania, mogą oznaczać zupełnie inne byty.

Pojęcie metafory nie jest zupełnie oderwane od rzeczywistości – ma również całkiem materialny wymiar. Metafory programistyczne podlegają patentowaniu (np. HyperCard – wczesny notatnik firmy Apple) i mogą być przedmiotem sprzedaży, nadużycia praw autorskich i zajadłych sporów prawnych.

8. Badanie metafor

Wraz z ugruntowaniem semantycznej analizy oprogramowania, wyodrębnia się coraz bardziej teoretyczny nurt tej dyscypliny. Stawia on sobie za cel tworzenie abstrakcyjnych systemów metafor, opisujących jak największą część pola zainteresowania geoinformatyki (jednym z wczesnych przykładów jest praca W. Kuhna i A.U. Franka z 1991 r.). Od końca lat dziewięćdziesiątych jest w użyciu pojęcie odwzorowania semantycznego. Zostało ono zdefiniowane pod nazwą *semantic mapping* przez G. Fauconniera (1997), a rozwinięte i sformalizowane pod nazwą *semantic morphing* przez J. Gougena (1999). Metoda ma zastosowanie w sztucznej inteligencji i lingwistyce (C.L. Nehaniv 2000) oraz analizie metafor w płytkiej komunikacji z użytkownikiem (J. Gougen 2003).

Teoretyczne osiągnięcia mają pewien wpływ na praktykę GIS, dzieje się to jednak z dużym opóźnieniem i wybiórczo. Należy wątpić, aby którakolwiek z teorii mogła w całości służyć za podstawę budowy nowego systemu informacji geograficznej, trudno również znaleźć taką, która byłaby stworzona specjalnie z myślą o ocenie i porównaniach istniejących programów. Aby zrealizować cel postawiony na początku artykułu, a więc uzyskanie nowego, pogłębionego spojrzenia na istniejące programy GIS, trzeba wykorzystać mniej rygorystyczny wariant analizy semantycznej.

Model postępowania w badaniu metafor można wzorować na procesie uczenia się metafor przez użytkownika, który poznaje nieznane oprogramowanie. Rozpoczyna się on od poznania nazwy, która ma jakieś odniesienie do świata realnego i przywołuje bagaż doświadczeń, skojarzeń i oczekiwań. Potem następuje rozpoznanie właściwości, które będą w pewnym stopniu zgodne z oczekiwaniami, a w pewnym – nie. Kiedy użytkownik dowiaduje się, że w systemie funkcjonuje metafora mapy, może się spodziewać, że mapę tę można oglądać i drukować, natomiast ze zdziwieniem przyjmie informację, że mapa może mieć praktycznie nieskończony rozmiar lub że można płynnie zmieniać jej skalę. Przedmiot realny, do którego odwołuje się metafora mapy komputerowej, nie ma takich właściwości. Jest zupełnie naturalne, że metafora jest obdarzona nowymi, nieznanymi właściwościami, które nie występują w świecie realnym – jej rolą jest przybliżyć abstrakcyjny byt, a nie zastąpić go.

Nowi użytkownicy starają się szybko dowiedzieć, co można zrobić przy użyciu metafory (czynny aspekt zachowania), dzięki czemu mogą ocenić jej wagę w systemie. Poznanie operacji, które można wykonywać w odniesieniu do metafory (bierny aspekt zachowania), jest kolejnym krokiem postępowania.

W dalszej kolejności użytkownik dowiaduje się, że metafora jest złożona lub zawiera inne metafory. Istotne są również powiązania, które łączą metaforę z innymi metaforami występującymi w systemie. Często wyjaśnienie budowy lub relacji prowadzi z lepszym skutkiem do zrozumienia znaczenia metafory, niż próba jej zdefiniowania. Szczególnym przypadkiem takich powiązań są czynności, które można wykonać za pomocą metafory czynnościowej w odniesieniu do metafory rzeczowej.

Analogicznie do procesu uczenia się, postępowanie badawcze powinno polegać na:

1) rozpoznaniu istotnych metafor (co samo w sobie może być trudnym zadaniem), a następnie

2) badaniu ich w następujących czterech aspektach:

– analogu: materialnego przedmiotu lub abstrakcyjnego obiektu, znanego ze świata realnego, do którego odwołuje się metafora;

– właściwości: cech metafory, które można poznać lub zmieniać, ze szczególnym uwzględnieniem cech, które nie występują w przedmiocie odniesienia;

– zachowania: czynności, które możemy wykonać w odniesieniu do metafory lub przy jej użyciu;

– budowy, jeśli w ich skład wchodzą inne metafory;

– relacji: sposobu powiązania z innymi metaforami występującymi w systemie.

Analiza semantyczna może być zastosowana do jednoczesnego, porównawczego badania wielu pakietów GIS. Można przypuszczać, że efektem będzie nie tylko pogłębienie wiedzy na temat oprogramowania, ale również odpowiedź na pytanie: ile z pojęć tradycyjnej geografii znalazło kontynuację w geoinformatyce i jakim uległy przekształceniom?

Literatura

Anderson A. A., 1996, *Predictors of computer anxiety and performance in information systems*. „Computers in Human Behavior" Vol. 12, no. 1, s. 61–77.

Booch G., Rumbaugh J., Jacobson I., 1999, *UML przewodnik użytkownika*. Warszawa: WNT, 2001.

Brodaric B., Gahegan M., 2002, *Distinguishing instances and evidence of geographical concepts for geospatial database design*. „Lecture Notes in Computer Science" Vol. 2478, s. 22–37. Berlin: Springer Verlag.

Chalmers P. A., 2003, *The role of cognitive theory in human-computer interface*. „Computers in Human Behavior" Vol. 19, no. 5, s. 593–607.

Eco U., 1968, *Nieobecna struktura*. Warszawa: Wydawnictwo KR, 1996.

Fauconnier G., 1997, *Mappings in thought and language*. Cambridge University Press.

Goguen J., 1999, *An Introduction to algebraic semiotics, with applications to user interface design*. „Lecture Notes in Artificial Intelligence" Vol. 1562, s. 242–291. Berlin: Springer Verlag.

Goguen J., 2003, *Semiotic morphisms, representations, and blending for user interface design. Keynote lecture*. W: AMAST Workshop on Algebraic Methods in Language Processing, Proceedings. Verona: AMAST Press.

Gould M.D., McGranaghan M., 1990, *Metaphor in geographic information systems*. W: Proceedings, 4th Conference on Spatial Data Handling – Zurich 1990.

Kuhn W., Frank A.U., 1991, *A formalization of metaphors and image-schemas in user interfaces*. W: Frank A.U., Mark D.M. (Eds.) Cognitive and Linguistic Aspects of Geographic Space. Kluwer Academic Publishers.

Kuhn W., 2002, *Modeling the semantics of geographic categories through conceptual integration*. „Lecture Notes in Computer Science" Vol. 2478. Berlin: Springer Verlag.

Little M. R., Nigay L. (ed.), 2001, *Engineering for human-computer interaction*. Berlin: Springer Verlag.

Lackoff G., Johnson M., 1980, *Metafory w naszym życiu*. Warszawa: Państwowy Instytut Wydawniczy, 1988.

Montello D. R., Fabrikant S. I., Ruocco M., Middleton R.S., 2003, *Testing the first law of cognitive geography on point-display spatializations*. International Conference, COSIT 2003: Spatial Information Theory. „Lecture Notes in Computer Science" Vol. 2825, s. 316–331. Heidelberg: Springer Verlag.

Nehaniv C. L., 2000, *The making of meaning in societies: semiotic & information-theoretic background to the evolution of communication*. AISB Symposium: Starting from Society – the application of social analogies to computational systems. Proceedings. Society for the Study of Artificial Intelligence and Adaptive Behaviour, s. 73–84.

Rauterberg M., Hof M., 1995, *Metaphor engineering: a participatory approach*. W: Schuler W. i inni (Eds.), Designing User Interfaces for Hypermedia, s. 58–67. Berlin: Springer Verlag.

Richards J.R., Egenhofer M.J., 1995. *A comparison of two direct-manipulation GIS user interfaces for map overlay*. „Geographical Systems" Vol. 2, no. 4, s. 267–290.

Walker M.A., Whittaker S.J., Stent A., Maloor P., Moore J., Johnston M., Vasireddy G., 2004, *Generation and evaluation of user tailored responses in multimodal dialogue*, „Cognitive Science" Vol. 28, no. 5, s. 811–840.

Źródła internetowe

Apple Computer Inc., 2002, *Aqua human interface guidlines*. http://developer.apple.com/documentation

Costabile M.F., Fogli D., Letondal C., Mussio P., Piccinno A., 2003, *Domain-expert users and their needs of software development*. Session on End-User Development held at HCI International 2003 Conference. Proceedings EUD-Net. http://giove.cnuce.cnr.it/EUD-NET/pdf/D3.4.pdf

Microsoft Corporation, 2001, *Microsoft inductive user interface guidlines*. http://www.microsoft.com

Recenzował dr hab. Piotr Werner

Semantic analysis of GIS software

Summary

Keywords: GIS theory, geoscience, software engineering, semiotics, cognitive science, metaphor

GIS software descriptions are widespread but often useless, because they are unable to reveal the inner logic of the system. Like other complex software, GIS may be analysed in terms of three layers: 1) functionality, 2) user interface (shallow layer) and 3) cognitive structures (deep layer). Vendor information sources, factsheets and application studies cover, for the most part, functionality and shallow layer, neglecting deep structure. However, misunderstanding and mismatch between user cognitive structure and software structure severely impairs GIS usefulness, especially when big projects are concerned.

The rescue may come from cognitive science and semiology rather than from software engineering. Semantic analysis should be used to 1) achieve good design for new systems, 2) describe exsisting software and make a foundation for professional critical activity, similar to literary or architectural one.

Semantic analysis focuses on the mapping between domain-specific cognitive structure and corresponding deep software structure. The letter must be consistent (logically congruent), compact (devoid of unnecessary terms) and abstract (make use of general terms). Metaphors constitute basic components of deep software layer, conveying meaning and analogy from the real world or domain. To properly analyse system at hand, metaphors must be extracted first. The next step is to review each metaphor along five dimensions: 1) analogue (reference to external object), 2) properties, 3) behaviour, 4) composition (if consisting of simpler parts) and 5) relations to other metaphors within the system.

Translated by author

Семантический анализ программного обеспечения GIS

Резюме

Отсутствует метод описи программ GIS, который давал бы возможность познать и оценить не только их функции и способ коммуникации с потребителем, но также их внутреннюю логику. Программное обеспечение GIS можно анализировать на трёх уровнях: 1) функций, которые осуществляет, 2) организации работы и способа коммуникации с потребителем и 3) познавательных структур. Информации, представляемые производителем, а также работы, посвящённые практическим применениям GIS, охватывают лишь два первые уровни.

Третий, глубокий уровень программного обеспечения, требует также подлинного описания. Структура программы должна отражать структуру знаний географа или картографа и использовать упроченные в науке концепции. Если это не так, то пригодность GIS, особенно для проведения больших проектов, является низкой.

Семантический анализ, который функционирует уже давно в литературной, архитектурной критике и в семиологических исследованиях культуры, может быть успешно применён в отношении программного обеспечения, в том GIS. Он может служить как для описания существующих, так и проектирования новых систем.

Анализ должен отвечать на вопрос, является ли глубокая структура программного обеспечения логичной, сплочённой и абстрактной (общей). Следует выделить метафоры – основные элементы глубокой структуры, которые несут значение потребителю. Затем следует исследовать их под углом: 1) аналога (объекта относимости), 2) свойств, 3) поведения, 4) строения и 5) реляции.

Перевод Р. Толстикова