



Wywiad z Januszem Górskim

rozmawia Bartosz Chrabski

Czy inżynieria oprogramowania nadal potrafi nas naprawę zaskoczyć nowymi odkryciami i podejściami do już znanych od lat problemów?

Celem inżynierii oprogramowania jest skuteczne rozwiązywanie, przy zastosowaniu środków z dziedziny technologii informacyjnych, problemów z praktycznie dowolnych dziedzin aplikacyjnych. A więc na przykład rozwiązanie problemu ułatwienia dostępu klientów do usług bankowych, niezależnie od ich lokalizacji geograficznej względem siedziby i oddziałów tego banku. Rozwiązanie polega tu na wdrożeniu bankowości internetowej. Jeżeli zgodzimy się co do takiego rozumienia inżynierii oprogramowania, to zaskoczenie, o które Pani pyta, może pojawić się z dwóch źródeł: od strony problemów, które są przedmiotem zainteresowania, oraz od strony środków, które są wykorzystywane podczas rozwiązywania tych problemów.

Tendencje po stronie problemów to przede wszystkim poszerzający się ich zakres, zmienność, presja czasu oraz otwartość.

Poszerzanie zakresu oznacza, że rozwiązania informatyczne są stosowane w nowych obszarach i do problemów, które nie były wcześniej objęte informatyzacją. Zmienność oznacza, że identyfikacja problemu praktycznie nigdy się nie kończy – trzeba liczyć się z tym, że możemy jedynie lepiej lub gorzej nadążać za zmianami. Odpowiedzią na to jest położenie większego nacisku na współpracę z udziałowcami reprezentującymi dziedzinę problemową oraz utrzymywanie ciągłości tej współpracy w pełnym cyklu życia oprogramowania. Presja czasu powoduje, że inżynieria oprogramowania szuka metod umożliwiających szybką dostawę rozpoznawalnej dla klienta wartości dodanej, nawet jeżeli będzie ona dostarczana w sposób niepełny i potem uzupełniana kolejnymi przyrostami. Te dwie ostatnie tendencje są odzwierciedlone w tzw. podejściach „zwinnych”. Otwarcie na integrację oznacza, że powstające rozwiązania techniczne nie mogą być „zamknięte” i użytkowane w izolacji. Dynamika w świecie biznesu, w sferze społecznej czy w gospodarce skutkuje potrzebą integrowania istniejących i nowych systemów.

W wymienionych wyżej obszarach bardziej niż o rewolucji trzeba raczej mówić o ewolucji, ale ewolucji, która ostatnio gwałtownie przyspieszyła.

Wśród tendencji po stronie środków metodologicznych i technicznych można wyróżnić położenie większego nacisku na ciągły kontakt z kluczowymi udziałowcami, zarządzanie projektem

w sposób umożliwiający „zwinne” nadążanie za zmianami, architektury SOA i nacisk na integrację procesów biznesowych oraz poszukiwanie abstrakcji umożliwiających reprezentowanie oprogramowania w sposób niezależny od jego technicznej realizacji. W coraz większym stopniu rozpoznawane jest również znaczenie czynnika ludzkiego, zarówno po stronie analizy i oceny systemów, jak i po stronie wytwórczej.

W mojej ocenie po stronie technologicznej jest większa szansa na zaskoczenie. Technologie informacyjne już nie raz dowiodły, że ich rozwój może przyczynić się do zmian o charakterze rewolucyjnym. Przyczyną jest to, że mogą one otwierać nowe i nieprzewidziane wcześniej możliwości w dziedzinach problemowych, tak jak Internet otworzył drogę do nowych modeli świadczenia usług i w znacznym stopniu uwolnił biznes od ograniczeń geograficznych, a telefonia komórkowa zrewolucjonizowała strukturę komunikacyjną w wymiarze indywidualnym i biznesowym. Takie zmiany w infrastrukturze inspirowały nowe i gwałtownie rozwijające się potrzeby, a jednocześnie tworzą warunki do nowych i bardziej skutecznych sposobów zaspokajania tych potrzeb. Co będzie takim „zapalnikiem” w najbliższej przyszłości? SOA i integracja procesów biznesowych? *Cloud computing* jako nowy i uniwersalny model dostępu do usług IT? A może będzie to coś, co się „narodzi” na przecięciu informatyki i biotechnologii? Trudno przesądzać, tym bardziej że ludzie wielokrotnie już dowiedli, że przewidywanie przyszłości nie bardzo im się udaje. Ja przewiduję przesuwanie się inżynierii oprogramowania w stronę „inżynierii systemów” i związany z tym rosnący rynek pracy dla specjalistów łączących kompetencje z dziedzin aplikacyjnych z umiejętnościami obszaru analizy, (re)inżynierii procesów biznesowych oraz architektury oprogramowania i infrastruktury komunikacyjnej. Natomiast kompetencje odnoszące się do wytwarzania oprogramowania w sensie „umiejętności programowania” będą nie tyle zanikać, co lokalizować się w miejscach, gdzie takie oprogramowanie będzie tworzone. Czy zmiany te przybiorą charakter rewolucyjny, czy będzie to ewolucja – zobaczymy.

Czy w ostatnich latach powstały jasno identyfikowalne trendy w dziedzinie inżynierii oprogramowania, na które warto jest zwracać uwagę?

Niewątpliwie jest ich kilka. Wszystkie one w jakiś sposób narastają stopniowo, a więc ich identyfikacja jest bardziej ekstrapolacją historii

i stanu obecnego niż wizją jakiejś nowej zaskakującej zmiany. Przewiduję tu narastającą tendencję do poszerzania perspektywy w postrzeganiu oprogramowania w wymiarze systemowym (a więc bardziej *systems engineering* niż *software engineering*) i w związku z tym większy nacisk na poza-funkcjonalne własności związane z gwarancjami takimi jak bezpieczeństwo (*safety*), zabezpieczenie (*security*), prywatność czy zdolność systemów do przeżycia, pomimo niesprzyjających warunków wewnętrznych czy zewnętrznych. W tendencję tę wpisuje się również większa koncentracja na użytkownikach i dostarczanej im wartości dodanej. Spowoduje to rosnący nacisk na przekraczanie barier w komunikacji i współpracy między reprezentantami różnych dziedzin i organizacji oraz rosnące znaczenie pracy grupowej. Szybkość i zakres zmian oraz konieczność nadążania za zmianą utrzymują zainteresowanie metodami „zwinnymi” oraz traktowanie przedsięwzięć informatycznych jako procesów, które bardziej nadają się za „ruchomym celem” niż są realizowane według przygotowanego wcześniej szczegółowego planu. Utrzyma się tendencja do integracji i związany z tym wzrost złożoności systemów. Z jednej strony wymagać to będzie środków technicznych wspomagających taką integrację, z drugiej zaś strony niezbędne będą abstrakcje i metody umożliwiające przygotowanie tworzonych systemów do przyszłej integracji oraz zapanowanie nad wynikającą stąd złożonością. Wiąże się z tym również zagadnienie gotowych komponentów (*COTS – Commercial off-the-shelf*), czy szerzej, wielokrotne wykorzystanie istniejących już komponentów (*software re-use*). Rosnącego znaczenia będzie nabierać integracja systemów (*systems-of-systems*) nie tylko w wymiarze funkcjonalnym, ale również (a może nawet bardziej) w wymiarze związanych z nimi gwarancji pozafunkcjonalnych (takich jak bezpieczeństwo, dostępność czy niezawodność). Integracja ta będzie musiała również uwzględniać istnienie oprogramowania „odziedziczonego” (*legacy systems*).

Jakie obszary według Pana są najmniej zbadane i wymagają ciągłej pracy nad analizą tej dziedziny?

Pole do badań jest bardzo obszerne. W końcu mamy do czynienia z dziedziną, która nie „żyje” jeszcze nawet jednego pełnego stulecia. Której zakres oraz spektrum możliwych zastosowań są wciąż dalekie od pełnej identyfikacji. Która wciąż nie ma wspólnego i powszechnie akceptowanego systemu pojęć i związanego z nim języka. Nie

do końca wiemy, jak powtarzać sukces w budowie oprogramowania, nie rozumiemy i nie uzgodniliśmy kryteriów dla takiego sukcesu, mamy do czynienia z budulcem, który jest ulotny i może zmieniać swoje własności, nie mamy satysfakcjonującej wiedzy na temat procesów, w ramach których powstaje i jest eksploatowane oprogramowanie. Obszarów tych jest tak dużo, że trudno je wszystkie wymienić. Stan dziedziny jest w jakiś sposób odzwierciedlony w corocznych raportach *Standish Group*, które podają statystyki dotyczących udanych i nieudanych przedsięwzięć informatycznych. Dwie obserwacje w stosunku do tych danych są uderzające: (1) w pełni udanych jest około 30% przedsięwzięć (inne mają mniejsze lub większe trudności) oraz (2) proporcje te nie podlegają znaczącej zmianie wraz z upływem czasu. W jakiś sposób mówi to o stanie inżynierii oprogramowania i wskazuje na ogrom problemów czekających rozwiązania.

Osobiście uważam, że potrzebna jest eksploracja obszaru pomiędzy dwoma zauważalnymi tendencjami w podejściu do inżynierii oprogramowania: z jednej strony nadawanie priorytetu metodom i narzędziom, a z drugiej ludziom i ich profesjonalizmowi. Kierunki te w jakiś sposób odzwierciedlają różnice pomiędzy „sztywnym” i „zwinnym” podejściem do wytwarzania oprogramowania. Wydaje mi się, że bardzo ważna jest głębsza eksploracja i zrozumienie obszaru, który mieści się pomiędzy tymi skrajnościami.

Inny obszar, który mnie szczególnie interesuje, to rola zaufania i środki budowy zaufania w odniesieniu do procesów i produktów związanych z oprogramowaniem. Ma to silne przełożenie na takie własności poza-funkcjonalne jak bezpieczeństwo, zabezpieczenie czy prywatność. Wspólnym mianownikiem, który łączy te obszary, jest zarządzanie ryzykiem.

Oba wymienione wyżej obszary są przedmiotem badań w kierowanej przeze mnie grupie badawczej, która działa w Katedrze Inżynierii Oprogramowania Politechniki Gdańskiej.

Czy możliwe jest uporanie się z budową nowoczesnych, skomplikowanych systemów IT bez odpowiedniego warsztatu narzędzi?

Jak w każdej innej dziedzinie, narzędzia są potrzebne i mogą okazać się nadzwyczaj przydatne. Szczególnie gdy rozpatrujemy problemy skali, czy to w odniesieniu do złożoności, czy do wydajności lub kosztu. Narzędzia mogą również skutecznie chronić człowieka przed popełnianiem błędów i zdecydowanie zwiększają szansę na powtarzalność rezultatów. Nie wyobrażam sobie, by skuteczne podjęcie wyzwań, o których wspominałem wcześniej, było możliwe bez wsparcia narzędziowego. Należy jednak pamiętać, że programy budują ludzi, a nie metody i narzędzia. Oznacza to w szczególności, że narzędzi nie można rozpatrywać w izolacji od używających ich ludzi oraz zadań i procesów, w które są oni zaangażowani.

Jakie wyzwania według Pana stają przed nowoczesnymi narzędziami wspierającymi procesy twórczy na rynku?

Oczywistym wydaje się postulat, by narzędzia wspierały osiąganie celów, do których dążą ich użytkownicy. Tak więc cele wydają się być nadrzędne nad narzędziami, tzn. narzędzie, które nie wspiera zidentyfikowanego celu, jest bezużyteczne. Jednak związek ten nie jest aż tak wyraźnie jednokierunkowy. Zauważmy, że nie zwykle użyteczne narzędzie, młotek, przydaje się w bardzo wielu zastosowaniach, niekoniecznie oczywistych wtedy, gdy narzędzie to powstało po raz pierwszy. Istnieje więc jeszcze inne kryterium oceny narzędzi, które można by nazwać ich uniwersalnością, a więc zdolnością do użycia w różnych celach, niekoniecznie z góry przewidzianych. Uważam to za bardzo ważne, gdyż chroni to narzędzie przed moralnym starzeniem się i zwiększa jego zdolność do „przeżycia”. W przeciwieństwie do narzędzi zbyt wyspecjalizowanych, które się szybko dezaktualizują. Na gruncie informatyki, ze względu na zmienność w tej dziedzinie, istnieje wcale pokaźne „cementarysko” takich narzędzi.

Kolejny dylemat, to złożoność funkcjonalna. Przez analogię, czy lepiej używać młotka z końcówką do wyciągania gwoździ, czy też używać młotka „konwencjonalnego” i obcęgow? W pierwszym wypadku mamy dwie funkcje złożonym narzędziu, w drugim dwa narzędzia, każde do realizacji jednej prostej funkcji. Zauważmy, że w tym wypadku odpowiedź zależy od tego, jak wygląda proces użytkowania tych narzędzi. Bez analizy tego procesu trudno jest dać jednoznaczną odpowiedź. Jednak z punktu widzenia przyszłej ewolucji, zmian funkcji już istniejących lub dodawania nowych, rozwiązanie drugie, w duchu *unixowym*, wydaje się mieć przewagę. Tym bardziej podkreśla to konieczność zadbania o to, by narzędzia mogły ze sobą współpracować. A to z kolei zwraca uwagę na standaryzację dotyczącą takiej współpracy. I na konieczność właściwego rozumienia pojęcia „standard”. W pojęciu tym kryje się uzgodnienie i akceptacja, bez których „standard” standardem nie jest. Na tym tle dość chwiejne są tezy o tym, że nowo pojawiające się na rynku narzędzie „wprowadza standard” w dotyczącym go zakresie. Przy braku powszechnej akceptacji okres życia takich „standardów” jest czasem zaskakująco krótki.

Uniwersalność nie pozostaje w sprzeczności ze specjalizacją, jeżeli w architekturze narzędzia wyraźnie oddzielimy to, co uniwersalne, od tego, co wyspecjalizowane. W części wyspecjalizowanej można dopuścić do wyraźnego ukierunkowania na określoną metodę, funkcjonalność czy sprofilowanie użytkowników. Rozróżnienie takie wspomaga przystosowywanie narzędzia do zmian i w efekcie jego przeżywalność na rynku.

Narzędzi, za wyjątkiem być może tych najbardziej uniwersalnych, nie można rozpatrywać

w oderwaniu od kontekstu wspierającego ich rozwój i zastosowania. Wielokrotnie obserwowałem narzędzia (i związane z nimi metody), które szybko obumierały, gdy kończyły się związane z nimi wsparcie. Oprócz tych kryteriów ogólnych jest oczywiście bardzo ważną sprawą, by narzędzia odpowiadały na aktualne potrzeby. Niektóre z tych potrzeb wynikają z odpowiedzi, które udzieliłem na poprzednie pytania.

No i jeszcze jedna sprawa, którą uważam za ważną. Ze względu na rosące znaczenie pracy grupowej i rozproszenie użytkowników, uważam, że planując rozwój narzędzi, trzeba brać pod uwagę ich wersje „internetowe” z cienkim lub pogrubionym klientem. Niewpisanie się w ten trend grozi utratą rynku lub zdecydowanym jego zawężeniem. Warto również poważnie rozważyć model biznesowy polegający na udostępnianiu usług związanych z narzędziami, a nie model zakładający sprzedaż narzędzi jako produktów. Ten sposób myślenia o narzędziach może okazać się bardzo trafny w kontekście (być może) nadciągającej rewolucji związanej z *cloud computing*.

Janusz Górski (prof. dr hab. inż., prof. zw. PG) kieruje Katedrą Inżynierii Oprogramowania na Politechnice Gdańskiej. W pierwszej połowie lat 90. utworzył pierwszą w kraju specjalizację z zakresu inżynierii oprogramowania na studiach wyższych. W drugiej połowie lat 90. utworzył pierwsze w kraju studium podyplomowe inżynierii oprogramowania. W roku 1999 był inicjatorem i pierwszym przewodniczącym Krajowej Konferencji Inżynierii Oprogramowania, która do chwili obecnej jest główną platformą grupującą badaczy tej tematyki w kraju. Prowadził i konsultował wiele projektów rozwojowych i badawczych zarówno w kraju, jak i na forum międzynarodowym, w tym również w programach ramowych UE (programy ENVIRONMENT, COPERNICUS, 5. i 6. Programy Ramowe). Jest ekspertem Komisji Europejskiej w 5. 6. i 7. Programach Ramowych. Jest również doradcą Dyrektora European Network and Security Agency (ENISA) w zakresie kierunków rozwoju Agencji oraz oceny wybranych obszarów badawczych. Jego obecne zainteresowania dotyczą inżynierii oprogramowania (w szczególności problemów związanych ze skutecznym pozyskiwaniem oprogramowania przez klientów) oraz zarządzania ryzykiem dotyczącym procesów i produktów programistycznych. W szczególności w odniesieniu do ryzyka związanego z bezpieczeństwem, zabezpieczeniem i prywatnością. W kierowanym przez niego zespole (<http://iag.pg.gda.pl>) rozwijana jest innowacyjna w skali międzynarodowej metodologia Trust-IT, ukierunkowana na wspomaganie zarządzania argumentacją i wykorzystanie argumentów w budowie zaufania i w innych obszarach zastosowań. Jednym z takich obszarów jest wspomaganie procesów dochodzenia i oceny zgodności z normami i standardami (projekt NOR-STA, www.nor-sta.eu realizowany w ramach Programu Operacyjnego Innowacyjna Gospodarka).