Ryzyko etyczne w procesie wytwarzania oprogramowania

Maria Ganzha[*], Stanisław Szejko[**]

[*]*Katedra Informatyki, Prywatna Wyższa Szkoła Zawodowa w Giżycku*

[**]*Katedra Zastosowań Informatyki, Wydział Elektroniki, Telekomunikacji i Informatyki, Politechnika Gdańska*

ganzha@karolex.com.pl stasz@eti.pg.gda.pl

Streszczenie

W rozdziale rozpatrywane są współczesne problemy wytwarzania oprogramowania. Jednym z możliwych sposobów ich rozwiązywania jest uwzględnienie w procesie wytwarzania oprogramowania zagadnień ryzyka etycznego, rozumianego jako zagrożenia, jakie dla sukcesu przedsięwzięcia stwarzają problemy etyki podejmowanych decyzji, odstępstwa od obowiązującego prawa, uwarunkowań polityki prowadzenia projektu, wpływu projektu na ludzi, biorących w nim udział i otaczających go. Autorzy przytaczają niektóre ze współczesnych metod obniżania ryzyka etycznego i sugerują kierunki dalszego ich rozwoju.

1. Problemy wytwarzania oprogramowania

Na naszych oczach w ciągu ostatniego półwiecza dokonuje się rewolucja spowodowana rozwojem informatyki i jej zastosowań. Systemy informatyczne uczestniczą w zarządzaniu przedsiębiorstwami, służą komunikacji, biorą udział w sterowaniu obiektami od kosmosu po zarządzanie sprzętem domowym, są nieodłącznym elementem edukacji i zabawy – ich zastosowania wydają się nie mieć ograniczeń.

Rosną też wymagania im stawiane – oczekiwane są systemy o coraz większym poziomie złożoności, lepszym dopasowaniu do potrzeb, pełniejszej funkcjonalności, wyższej jakości i efektywności działania. Wszystko to powinno dziać się przy zwiększonej produktywności i malejących kosztach wytworzenia i utrzymania systemów.

Jednak bez względu na rosnący poziom wymagań i nowoczesne metody wytwarzania oprogramowanie wciąż „zawodzi". Sytuację pogarszają zmiany wymagań, które zachodzą nawet długo po wytworzeniu oprogramowania. Często proces wytworzenia oprogramowania kończy się niepowodzeniem w postaci niedostarczenia oczekiwanego produktu albo dostarczenia produktu o niepełnej funkcjonalności. Najgorsze jest to, że wady produktu często wychodzą na jaw w trakcie pracy gotowego produktu. Potwierdza to ciągły wzrost kosztów prac utrzymaniowych oprogramowania, a przy ich realizacji zatrudnionych jest ponad 60% informatyków. Trzeba jednak przyznać, że jest wiele przykładów oprogramowania wytworzonego na czas, bez przekraczania ograniczeń bu-

dżetowych, ze spełnieniem wymagań użytkownika, lecz codzienna praca dostarcza więcej problemów niż rozwiązań.

Uzasadnione jest więc pytanie: co może zmienić taką sytuację procesu wytwarzania oprogramowania? Wskazać można kilka dróg poszukiwania rozwiązań [BOEH2000, GORS2000, PMIS1996, SOMM1992, SZE2002a]:

- Poszukiwanie „lepszych" cykli wytwarzania. Cykl życia oprogramowania reprezentuje powtarzającą się w czasie całość działań prowadzonych od ujawnienia potrzeby budowy systemu do zakończenia jego wykorzystania. Szeroko znanych jest kilka modeli cyklu życia, które zdobyły trwałe miejsce w informatyce: kaskadowy, z zapewnieniem jakości, spiralny, ewolucyjno-przyrostowy, prototypowy. Jednak problemy w sferze wytwarzania oprogramowania wymagają „lepszych" rozwiązań będących źródłem oprogramowania o wyższej jakości, zwiększających produktywność lub obniżających koszt jego utrzymania.
- Wprowadzenie nowych technologii i związanych z nimi nowych metodyk. Na przykład wprowadzenia pojęć obiektu i klasy obiektów spowodowało istotne zmiany w cyklu życia i wytwarzania systemów informatycznych, przede wszystkim umożliwiło prowadzenie głównych faz cyklu – analizy, projektowania i implementacji w oparciu o te same (i kongruentne) modele. Również techniki takie jak wielokrotnie użycie komponentów lub wzorców programowych prowadzą do poprawy jakości oprogramowania i – często – skrócenia czasy jego realizacji.
- Bogatsze zastosowanie istniejących metod i notacji, na przykład wykorzystanie metod formalnych do specyfikacji wymagań.
- Ulepszanie metod zarządzania projektami informatycznymi i procesami wytwórczymi, w tym – ludźmi i zespołami zaangażowanymi w projekt (projektowymi, realizacyjnymi).
- Rozwój i wykorzystanie środowisk wspierających i narzędzi niejednokrotnie integrujących różne sfery prowadzenia projektów.

Mimo intensywnych prac we wszystkich tych kierunkach, wiele powstających systemów nie spełnia oczekiwań użytkownika, wytworzenie oprogramowania wciąż trwa zbyt długo, a jego utrzymanie zbyt wiele kosztuje. Silnie manifestują się problemy naruszenia przepisów prawnych oraz implikacji społecznych, jakie niesie sam proces wytwórczy i późniejsze użytkowanie systemu.

Gotterbarn wskazuje dwa źródła omawianych problemów [GOT2001]:

- projektanci zbyt wąsko traktują pojęcie udziałowców projektu[1], poszukując ich głównie wśród użytkowników i bezpośrednich beneficjentów systemu. A to może prowadzić do wytworzenia systemu, który ma nieoczekiwane oddziaływania negatywne. Jako przykład można wspomnieć system radarowy Aegis (*Aegis Fire System Control Radar),* który „samodzielnie" zestrzelił samolot irańskich cywilnych linii lotniczych [WWW2003a]. Także nowy aparat do naświetleń THERAC-25 spełniał wszystkie zdefiniowane wymagania, lecz dopiero później okazało się, że był szkodliwy dla pacjentów, jako że od czasu do czasu generował zbyt dużą dawkę promie-

[1]Udziałowcami nazywane są osoby i grupy, które w sposób *bezpośredni* lub *pośredni* mogą zyskać albo też ponieść straty w wyniku pracy tworzonego oprogramowania.

niowania [WWW2003b].

- projektanci biorą pod uwagę zwykle tylko ryzyko techniczne (bezbłędnie napisany program) i ryzyko przekroczenia kosztów.

Stosowanym zwykle rozwiązaniem jest tu cykl spiralny wytwarzania oprogramowania zbudowany w oparciu o strategię minimalizacji ryzyka oraz wprowadzenie oceny przez użytkownika i czynności związanych z utrzymaniem przy każdym powtórzeniu cyklu [BOEH2000]. Model ten, a przede wszystkim jego rozwinięcie *WinWin,* zawiera fazę analizy ryzyka, która obejmuje ocenę rozwiązań alternatywnych oraz identyfikację i osza-cowanie związanego z nimi ryzyka (technicznego, ekonomicznego, organizacyjnego) realizacji nowej wersji systemu. W każdym kolejnym cyklu projektant musi zidentyfikować udziałowców systemu, ustalić „warunki wygrania" (ang. *win conditions*) dla każdego nowego udziałowca, a następnie ocenić czy nowe „warunki wygrania" pasują do zbioru „warunków wygrania", które ustalono dla poprzednio rozpatrywanych udziałowców.

Także inne metody starają się przewidywać zagrożenia i minimalizować ich negatywny wpływ na projekt. LIME [BUAB1999] wprowadza wielowymiarową analizę jakości wytwarzanego oprogramowania, Staged Model [RABE1999] dyskutuje projekt jako kompozycję faz, jednak żadna ze wskazanych propozycji nie rozpatruje ryzyka problemów etycznych, politycznych, czy odstępstw od przepisów prawnych w powiązaniu z wytworzeniem i użytkowaniem oprogramowania.

2. Analiza wpływu

Co zmieni się w cyklu wytwarzania oprogramowania w przypadku wprowadzenia analizy ryzyka etycznego (przy czym przez *ryzyko etyczne* rozumiemy tu zagrożenia, jakie dla sukcesu przedsięwzięcia projektowego stwarzają problemy etyki podejmowanych decyzji, odstępstwa od obowiązującego prawa, wpływ projektu na biorących w nim udział i otaczających go ludzi, uwarunkowania polityki prowadzenia projektu)?

Większość uznanych cykli wytwórczych ma wspólną fazę – fazę analizy. W ramach analizy ma miejsce:

- identyfikacja, gromadzenie i specyfikacja *szeroko rozumianych* wymagań użytkowych. Obejmują one m.in. cele biznesowe systemu, jego funkcjonalność, zakres informacji, na której system operuje, docelowe środowisko pracy systemu, cechy i ograniczenia użytkowe, oczekiwania i parametry jakościowe, wymagania i standardy względem procesu wytwórczego;
- modelowanie, obejmujące analizę zebranych wymagań i konstrukcję modelu logicznego systemu, którego działanie spełni te wymagania.

Bardzo naturalne byłoby rozszerzenie analizy wymagań o identyfikację wszystkich udziałowców i analizę ryzyka nie tylko ekonomicznego i technicznego, a również socjalnego, politycznego i etycznego.

Dotychczas proces identyfikacji udziałowców polega zwykle na określeniu wyłącznie bezpośrednich udziałowców, to znaczy tych, którzy są bezpośrednimi użytkownika-

mi systemu lub odnoszą bezpośrednią korzyść z jego wprowadzenia. Musimy rozszerzyć tradycyjną listę udziałowców o udziałowców pośrednich, czyli tych, na których oprogramowanie lub jego produkty będą oddziaływać. W takim przypadku lista udziałowców będzie zawierała użytkowników oprogramowania, ich rodziny, przedsiębiorstwa, których struktura może znacznie się zmienić po wprowadzeniu oprogramowania, środowisko, społeczeństwo, projektantów oprogramowania, pracowników firm wytwarzających oprogramowanie, same te firmy, ludzi, którzy zyskują na wprowadzeniu oprogramowania i którzy z tego powodu tracą, itp.

Tworzenie listy udziałowców zaczyna się od analizy wstępnej celów i zadań projektowanego systemu informatycznego. Proces ma charakter iteracyjny, bo każda konkretyzacja zadań systemu może wymagać kolejnego rozszerzenia listy udziałowców. Iteracje (czyli nawroty do listy udziałowców) mogą występować na każdym etapie. Proces ten jest częścią Software Development Impact Statement (SoDIS) [GOT2001]. W założeniu, proces SoDIS obejmuje cztery podstawowe kroki:

- identyfikację bezpośrednich i pośrednich udziałowców;
- identyfikację zadań, z których składa się projekt;
- zidentyfikowanie, dla każdego z tych zadań, prawdopodobnych problemów etycznych, które mogą powstać w trakcie wykonania zadania dla każdego z udziałowców;
- zaproponowanie rozwiązania – najczęściej będzie ono prowadziło do modyfikacji struktury zadaniowej (WBS, ang. *Work Breakdown Structure*) projektu.

Właśnie na takiej zasadzie działa program SoDIS Project Auditor, przedstawiony na odbytych w maju tego roku w Giżycku warsztatach „Professionalism in Software Engineering 2003" [SPAM2003]. Doświadczenie pracy z tym programem wskazuje, że wprowadzenie rozszerzonej listy udziałowców i analiza projektu z etycznego punktu widzenia uzasadnione są nie tylko z punktu widzenia etyczno-filozoficznego, ale również z punktu widzenia inżynierii oprogramowania. W trakcie pracy przeanalizowanych było kilka projektów. Najciekawsze jest to, że analiza etyczna z rozszerzoną listą udziałowców pozwoliła wykryć właśnie te wady projektów, które w swoim czasie stały się powodem niepowodzenia analizowanych systemów informatycznych.

W pracy [SZE2002b] zaproponowano rozwinięcie metody SoDIS. Uwzględnia ono rozdzielne istnienie współdziałających, synergicznych procesów Zamawiającego oraz Wytwórcy i postuluje dekompozycję włączanych do cyklu życia zadań zapewniania etyki i minimalizacji ryzyka pomiędzy te procesy.

Prowadzenie rozszerzonej analizy wymagań (analizy wpływu, ang. *impact statement analysis*) oczywiście wymaga od projektantów nowego spojrzenia na podejmowany projekt. Analiza wpływu metodą *SoDIS* „zmusza" projektantów do myślenia o ewentualnych udziałowcach projektu, o ich ewentualnych stratach i korzyściach i wiąże projekt z rzeczywistym, społecznym kontekstem, w którym oprogramowanie będzie wytworzone i ma działać.

3. Podejmowanie etycznych decyzji

Drugim typem sytuacji, na jakie napotyka uczestnik procesu projektowego, a należących do sfery ryzyka etycznego jest podejmowanie decyzji mających wymiar etyczny. Decyzje takie mogą naruszać czyjeś uprawnienia lub przywileje, zagrożenie popełnienia błędu przy ich podejmowaniu jest dość wysokie, a konsekwencje błędu dla społeczności, jednostki lub organizacji są zwykle bardzo znaczące.

W poszukiwaniu wzorców podejmowania takich decyzji Bynum i Rogerson proponują „heurystyczną metodę" prowadzenia analizy etycznej projektu [BYRO2003]. Zakłada się, że dysponujemy pełnym opisem wszystkich szczegółów projektu. Analiza przebiega w kilku krokach:

1. Przyjęcie etycznego punktu widzenia problemu.
   Wstępnym założeniem analizy jest akceptacja takich pojęć jak równość ludzi, sprawiedliwość i poszanowanie ludzkich praw.
2. Szczegółowy opis projektu.
   Przed dokonaniem analizy projektu bardzo ważne jest posiadanie jasnego i szczegółowego opisu rozpatrywanego przedsięwzięcia, zawierającego wszystkie szczegóły dotyczące projektu.
3. Identyfikacja problemów etycznych i ich rozwiązań.
   Po stworzeniu opisu szczegółowego należy zidentyfikować problemy etyczne, związane z projektem, a również ustalić czy istnieją odpowiadające im tradycyjne rozwiązania (bardzo wiele problemów etycznych ma już swoje rozwiązania).
4. Wykorzystanie własnej wiedzy i umiejętności etycznych.
   Jeżeli w trakcie analizowania projektu napotyka się nowy, nie rozpatrzony wcześniej przypadek lub chcemy dogłębniej zrozumieć sytuację, można wykorzystać którąś z poniższych metod:

- Wykorzystanie precedensów i analogii.
  Zastanowienie się nad ewentualnymi precedensami, analogiami, przykładami może pomóc rzucić światło na problem.

- Wykorzystanie własnej wrażliwości wobec problemu.
  Będzie to próba wyobrażenia sobie czy jest ktoś – ktokolwiek, – kto mógłby sprzeciwiać się danej sytuacji. Jest prawdopodobne, że ten ktoś odczuwa zagrożenie tą sytuacją lub faktem, że rozpatrywany problem dotyczy jego obowiązków i odpowiedzialności. Identyfikacja ewentualnych oponentów, jak też powodów ich sprzeciwów może pomóc skoncentrować się na kluczowych problemach etycznych. Można domniemywać, że te problemy stanowią główną trudność rozpatrywanego projektu.
- Postawienie się w położeniu udziałowca.
  Odpowiada to symulacji punktu widzenia udziałowca, podobnie jak to czasem robimy przy identyfikacji i analizie wymagań użytkownika.

5. Wykorzystanie wiedzy etycznej i etycznych umiejętności innych.
   Korzysta się tu głównie z konsultacji i doradztwa, ale także z gotowych rozwią-

zań – wzorców postępowania.

6. Wykorzystanie jednej lub kilku technik prowadzenia analizy.

- Analiza profesjonalna.
  Wykonywane czynności lub podejmowane decyzje są rozpatrywane pod kątem istniejących standardów i przyjętych unormowań kodeksowych, głównie kodeksu przyjętego przez ACM i IEEE dla inżynierii oprogramowania *Software Engineering Code of Ethics and Professional Practice* [ACIE1998] lub kodeksów innych uznanych organizacji, jak The British Computing Society, The Institute for the Management of Information Systems, The Australian Computing Society.
- Analiza ról i obowiązków.
  Ten typ analizy polega na systematycznym rozpatrywaniu ról udziałowców projektu.
- Wykonanie analizy udziałowców projektu.
  Systematycznie, dla każdego udziałowca, rozpatrywane są jego ewentualne straty lub korzyści, co może pozwolić lepiej zrozumieć problemy etyczne związane z projektem.
- Wykonanie „analizy etyczno-filozoficznej".
  Do analizy projektu wykorzystywane są główne idee wybranych i uznanych teorii etycznych, jak utylitaryzm, teorie Arystotelesa czy Kanta.

7. Wyciągnięcie etycznych wniosków.

Po wykonaniu wymienionych wyżej kroków można już odpowiedzieć na podstawowe pytania

- Jakie są kluczowe problemy etyczne?
- Czy ktoś postępuje nieetycznie?
- Czy ktoś będzie zmuszony postępować nieetycznie? itp.

8. Wyciągnięcie ewentualnej nauki na przyszłość.

Jeżeli ktoś z uczestników projektu postępował nieetycznie, w jaki sposób można zapobiec temu w przyszłości? Jeżeli problemy, dotyczące projektu nie były rozpatrzone wcześniej, w jaki sposób rozwiązywać je w przyszłości?

Innym rozwiązaniem mogłoby być oparcie rozwiązania problemów decyzyjnych o system *wzorców postępowania*[2], wspomagających identyfikacje i analizę problemów cyklu wytwórczego (cyklu życia) oraz wspierających podejmowanie decyzji. Projekt opracowania zestawu wzorców, wspomagających analizę i rozwiązanie infoetycznych problemów procesu wytwarzania oraz budowę prototypu ich biblioteki definiowany jest aktualnie na Wydziale ETI Politechniki Gdańskiej.

4. Podsumowanie

[2]Wzorce postępowania (działania) stanowią gotowe rozwiązania w sferze modeli celowej działalności, z których można skorzystać określając rozwiązanie zidentyfikowanych proble-mów [SZEJ2002a].

Dokonany przegląd sytuacji, a także los wspomnianych w pierwszej sekcji projektów, uznawanych początkowo za zakończone z sukcesem, a które pociągnęły za sobą niepożądane, wręcz tragiczne skutki, wyraźnie wskazują na potrzebę ujęcia zagadnień minimalizacji ryzyka etycznego w projektach informatycznych. Dokonać tego można poprzez:

- modyfikację – lub opracowanie nowych – istniejących cykli życia systemu o punkty identyfikacji typowych zagrożeń ryzyka, lub o metody rozpoznawania pojawienia się takich zagrożeń. Realizacja związanych z tym zadań wymaga starannego prześledzenia współczesnych cykli rozwojowych oprogramowania, wielu przypadków rzeczywistych, udanych i nieudanych przedsięwzięć projektowych oraz dogłębnej ich analizy pod kątem etyczno – prawnym;
- zaproponowanie metod i narzędzi wspomagających analizę identyfikowanych zagrożeń i dobór strategii postępowania oraz wspierających rozwiązanie problemów bądź minimalizowanie skutków zagrożeń. Trzy przykłady takich środków to rozszerzona analiza wpływu udziałowców, publikowane standardy i rozwiązania kodeksowe oraz biblioteka wzorców postępowania.

Bibliografia

[ACIE1998] *ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices, Software Engineering Code of Ethics and Professional Practice*, http://www.acm.org/serving/se/code.htm .

[BOEH2000] B. Boehm, *Spiral Development: Experience, Principles and Refinements*, Spiral Experience Workshop 2001, http://www.sei.cmu.edu/cbs/spiral2000/february 2000/Boehm/sld001.htm .

[BUAB1999] L. Buglione i L. Abran, *LIME: A Threee-Dimensional Measurement Model for Life Cycle Project Management*, Proc. of the International Workshop on Software Measurement, Sept, 1999.

[BYRO2003] T.W. Bynum i T.W. Rogerson, *Computer Ethics and Professional Responsibility*, Blackwell Publishing, 2003, in print, Part 1 What is Computer Ethics.

[GORS2000] J. Górski, *Inżynieria oprogramowania w projekcie informatycznym*, wyd. 2 rozszerzone. Mikom, 2000.

[GOTT2001] D. Gotterbarn, *Reducing Software Failures: Addressing the Ethical Risks of the Software Dewelopment Lifecycle*, Proc. of the 5th International Conference on The Social and Ethical Impacts of ICT ETHICOMP 2001, Gdańsk, June 2001, vol. 2, pp. 10 –19.

[PMIS1996] Standards Comittee PMI, *A Guide to the Project Management Body of Knowledge*, 1996.

[RABE2000] V.T. Rajlich i V.T. Bennett, *A Staged Model for the Software Life*

Cycle, IEEE Computer, July 2000, 66-71.

[SOMM1992] I. Sommerville, *Software Engineering*, Addison - Wesley, 1992.

[SPAM2003] *SoDID Project Auditor User's Manual (extract)*, Materiały Professionalism in Software Engineering Workshop, Giżycko, 7-9 maja 2003.

[SZE2002a] S. Szejko, *Metody wytwarzania oprogramowania*, MIKOM, Warszawa, 2002.

[SZE2002b] S. Szejko, *Incorporating Ethics into the Software Process*, Mat. VI konferencji ETHICOMP 2002 "The Transformation of Organisations in the Information Age: Social and Ethical Implications., Lizbona, Listopad 2002, str. 271 – 279.

[WWW2003a] *Strona internetowa*, http://www.cse.nd.edu/~kwb/nsf-ufe/safety-gotterbarn.html.

[WWW2003b] *Strona internetowa*, http://www.computingcases.org/case_materials/therac/therac_case_intro.html.