STUDIA OECONOMICA POSNANIENSIA 2013, vol. 1, no. 2(251)

Dawid Grzegorz Węckowski
Uniwersytet Ekonomiczny w Poznaniu, Wydział Informatyki i Gospodarki Elektronicznej, Katedra Informatyki Ekonomicznej
dawid.weckowski@kie.ue.poznan.pl

AUTOMATYZACJA DZIAŁAŃ UŻYTKOWNIKA W PRZEGLĄDARKACH INTERNETOWYCH WSPOMAGAJĄCA PROCESY BIZNESOWE

Streszczenie: Automatyzacja czynności wykonywanych w przeglądarce pozwala na usprawnienie pracy z aplikacjami internetowymi, zwłaszcza dla powtarzalnych działań wykonywanych przez internautów. Dzięki temu systemy automatyzacji mogą być wykorzystane przy wykonywaniu określonych elementów procesów biznesowych, wymagających interakcji z aplikacjami w sieci Web. W artykule przedstawiona została analiza możliwości automatyzacji powtarzalnych czynności użytkowników przeglądarek internetowych w ramach realizacji procesów biznesowych. Zaproponowano również zestaw poleceń automatyzacji uwzględniający standardowy zakres działań internauty.

Słowa kluczowe: automatyzacja, przeglądarka internetowa, procesy biznesowe.

Klasyfikacja JEL: L86.

Wstęp

Współczesny Internet nieustannie zmienia swe oblicze – z ogromnej kolekcji połączonych ze sobą statycznych dokumentów staje się siecią interaktywnych aplikacji i usług [Rossi i Turrini 2008]. Coraz częściej adresy stron internetowych nie wskazują na konkretne dokumenty HTML, ale prowadzą do treści tworzonych dynamicznie w odpowiedzi na żądanie HTTP. Dostęp do tych treści warunkowany jest między innymi indywidualnymi preferencjami użytkownika, zasobnością baz danych, z których pochodzą prezentowane informacje, czy też pomyślnością procesów uwierzytelniania i autoryzacji w ramach użytkowania aplikacji internetowych.

Wzrasta również złożoność interakcji użytkownika z odwiedzaną stroną – w wielu wypadkach konieczne jest zapamiętywanie i wprowadzanie identyfikatorów oraz haseł, wypełnianie rozbudowanych formularzy lub nawigacja przez określone adresy URL [Safonov, Konstan i Carlis 2001]. Przykładem tego typu działań może być sprawdzenie: cen biletów lotniczych, dostępności książek w bibliotece czy też uzyskanie szczegółowych danych na temat konkretnego pracownika w firmie.

W działaniach tych można odnaleźć powtarzalne sekwencje zdarzeń, które mogą być wykonywane automatycznie, dla podniesienia wydajności i niezawodności ludzkiej pracy [Bolin i in. 2005].

Szczególnego znaczenia nabiera automatyzacja pracy ze stronami należącymi do tak zwanego głębokiego Internetu [Calì i Martinenghi 2010], których nie identyfikuje jednoznacznie adres sieciowy, a które są dostępne jako wynik wykonania właściwej sekwencji czynności w przeglądarce internetowej. Aplikacje pozwalające na odtworzenie takich akcji muszą odpowiednio pozyskiwać oraz integrować dane ze stron internetowych, w ramach zdefiniowanego modelu dokumentu oraz zakresu interakcji użytkownika ze stroną. Należy również zwrócić uwagę, że działania użytkownika w przeglądarce mogą być częścią procesu biznesowego, którego realizacja nie jest ściśle zintegrowana z systemami automatyzacji [Rossi i Turrini 2008].

W artykule przedstawiono problematykę automatyzacji czynności wykonywanych przez użytkownika przeglądarki wraz z analizą możliwości wspierania realizacji procesów biznesowych w zakresie powtarzalnych zadań. Na podstawie dokonanego przeglądu istniejących rozwiązań w tym obszarze zaproponowano podejście do automatyzacji, które może zostać wykorzystane w definiowaniu i wykonywaniu procesów biznesowych angażujących aplikacje sieci Web i przeglądarki internetowe.

1. Problematyka automatyzacji czynności użytkownika w przeglądarce

Potrzeby automatyzacji czynności użytkownika i kierunki rozwoju w tym zakresie podyktowane są charakterystyką wykorzystania przeglądarki jako narzędzia prezentacji treści, jak również sposobami nawigacji w sieci oraz możliwościami rejestrowania czynności użytkowników.

1.1. Aktywność użytkownika przeglądarki

W pracy internautów daje się zauważyć pewną powtarzalność działań, wyrażającą się w wielokrotnym odwiedzaniu tych samych stron, w zadawaniu podobnych zapytań do często aktualizowanych źródeł danych lub w podążaniu podobnymi ścieżkami nawigacyjnymi w sieci [Adar, Teevan i Dumais 2009]. Użytkownik staje przed koniecznością znalezienia sekwencji czynności, która doprowadzi go do pomyślnego zakończenia postawionego sobie zadania. Następnie musi on odtworzyć ową sekwencję za każdym razem, gdy powtórnie wykonuje to zadanie lub inne o zbliżonym charakterze [Faaborg i Lieberman 2006].

Najpopularniejsze dziś przeglądarki mają wiele rozwiązań wspomagających użytkownika w odnajdowaniu i wykorzystywaniu treści stron internetowych. Stan-

dardem są obecnie przyciski nawigacyjne, system zakładek (ang. *bookmarks*) czy też gama rozszerzeń (ang. *extensions*) wspomagających szybsze wyszukiwanie bądź filtrowanie informacji [Abramowicz 2008].

Jak pokazują badania (tabela 1), do tej pory dominujące akcje nawigacyjne, takie jak użycie hiperłączy lub przycisku Wstecz, ustępują miejsca różnorodnym czynnościom, których nie można zidentyfikować, analizując zdarzenia w kodzie HTML lub standardowym interfejsie przeglądarki. Należą do nich zdarzenia wywołane przy użyciu elementów bogatego interfejsu użytkownika, wykorzystującego w budowaniu stron internetowych najnowsze technologie.

Tabela 1. Procentowy udział wybranych akcji nawigacyjnych podejmowanych przez użytkowników przeglądarek internetowych w latach 1995–2006

Rok	Akcja nawigacyjna		
	hiperłącze	przycisk Wstecz	niezidentyfikowana
1995	51,9	40,6	–
1997	50,0	30,0	–
2004	42,5	22,7	10,7
2006	41,0	18,9	23,6

Źródło: Opracowanie własne na podstawie: Catledge i Pitkow [l995], Kellar, Watters i Shepherd [2006], Milic-Frayling i in. [2004], Tauscher i Greenberg [l997].

Sposób, w jaki użytkownicy posługują się przeglądarką, a także rodzaj zadania, jakie mają do wykonania, wpływają na zakres potrzebnej automatyzacji. Wiele zależy jednak od budowy stron internetowych oraz możliwości rozpoznania akcji podejmowanych przez użytkownika.

1.2. Rejestrowanie zdarzeń w przeglądarce

Do niedawna powszechną metodą rejestrowania zdarzeń w przeglądarce była analiza logów serwera, które przechowują dane na temat żądań, otrzymywanych od przeglądarki użytkownika, oraz danych przesyłanych w odpowiedzi. Wśród innych popularnych sposobów można wymienić używanie pośrednika sieciowego (HTTP proxy) [Hong i Landay 200l], za pomocą którego przechwytywane są dane o stronach odwiedzonych przez użytkownika. Zwykle rozwiązania takie zbierają dane jedynie o odwiedzonych adresach URL, chociaż powstają również takie, które poprzez dodawanie odpowiedniego kodu JavaScript zapewniają przesyłanie większej ilości danych o zdarzeniach do pośrednika [Atterer, Wnuk i Schmidt 2006].

Połączenie kilku technologii w celu implementacji tzw. inteligentnych zakładek (ang. *smart bookmarks*) do stron, które nie są dostępne poprzez zwykły adres URL, ponieważ wymagane jest podjęcie dodatkowych działań, np. wysłania formularza, zaproponowano w systemie WebVCR [Anupam i in. 2000], wykorzystującym język

JavaScript wraz z apletami Java oraz LiveConnect[1]. Częstym sposobem rejestrowania czynności użytkownika jest również instalowanie dodatkowego oprogramowanie w postaci agenta programowego [Goecks i Shavlik 2000] lub dodatku do przeglądarki. Z takiego podejścia korzystają rozwiązania przedstawione w sekcji 2 artykułu.

Spośród pozostałych metod warto wspomnieć o używaniu skryptów uruchamianych przez przeglądarkę użytkownika, które śledzą akcje wskaźnika myszy [Mueller i Lockerd 2001], wiążą je z konkretnym czasem i wysyłają dane do serwera w celu dalszej analizy. Ciekawym podejściem jest dołączenie informacji o miejscu na ekranie, na które zwrócony jest wzrok użytkownika [Reeder, Pirolli i Card 2001].

Możliwości metod pozyskiwania informacji o czynnościach użytkownika w przeglądarce wpływają bezpośrednio na wyzwania, jakim muszą sprostać autorzy systemów automatyzujących te czynności.

1.3. Wyzwania dla twórców systemów automatyzujących nawigację w przeglądarce

Przed twórcami systemów automatyzujących nawigację w sieci Web stoi kilka kluczowych wyzwań, jak: obsługa częstych zmian zawartości i struktury stron internetowych, określenie stanu aplikacji, niejasność przekazywanej informacji i nowe technologie [Safonov, Konstan i Carlis 2001]. Zostaną one omówione poniżej.

Obsługa częstych zmian zawartości i struktury stron internetowych
Zmiany na stronach sieci Web powinny być interpretowane w sposób możliwie najbliższy zachowania człowieka i w jak najmniejszym stopniu wpływać na wynik procesu automatyzacji:

- odwołania zależne od kontekstu i pozycji na stronie: bardzo często występują rozbieżności pomiędzy zawartością strony w momencie wykonywania procesu automatyzacji a zawartością, która istniała w momencie tworzenia schematu procesu [Berendt i Spiliopoulou 2000]; na przykład, hiperłącze o niezmienionej nazwie może prowadzić do innego adresu URL lub może się zmienić kolejność hiperłączy wyświetlanych na stronie,
- niedające się odtworzyć ścieżki nawigacyjne: w celu dostarczenia dodatkowych informacji do serwera, na stronach zamieszczane są adresy URL i ukryte formularze, które zawierają wartości generowane losowo lub zależne od czasu; w związku z tym powtórzenie niektórych sekwencji nawigacji staje się niemożliwe.

Określenie stanu aplikacji

- Pliki *cookie*: zawartość strony zwróconej w odpowiedzi na żądanie użytkownika w wielu wypadkach jest uzależniona od informacji przechowywanych

[1] Więcej na temat tej technologii można znaleźć pod adresem internetowym http://www.mozilla.org/js/liveconnect/ [dostęp: 30.12.2012].

w plikach *cookie*; skuteczna automatyzacja musi uwzględniać zawartość tych plików podczas rejestrowania czynności użytkownika.
– Niepożądane efekty działań: metody protokołu HTTP zostały zdefiniowane w taki sposób, aby pojedyncze i wielokrotne wywołania każdej z nich dawały taki sam efekt, jednakże uruchamianie skryptów na serwerze może prowadzić do powstania dodatkowych efektów działań, na przykład w postaci obciążenia rachunku bankowego; poprawne określenie i wywoływanie takich efektów jest poważnym wyzwaniem dla systemów automatyzacji, gdyż skutki aktywności w przeglądarce nie zawsze mogą być precyzyjnie określone.

Niejasność przekazywanej informacji
W przeciwieństwie do człowieka, systemy automatyzacji nie są w stanie zrozumieć treści zawartych na stronach internetowych i wnioskować na ich podstawie, przez co implementacja odpowiednich automatycznych reakcji na zdarzenia w przeglądarce jest skomplikowana.
– Weryfikacja efektu nawigacji: człowiek może bez trudu stwierdzić, czy nastąpił błąd logowania i czy, ze względu na błędnie wprowadzoną nazwę użytkownika lub hasło, należy powtórzyć procedurę logowania; dla systemu, który ma automatyzować działania użytkownika, takie wnioskowanie nie jest oczywiste.
– Trudność w rozpoznawaniu zawartości: strony internetowe zawierają treści w wielu formatach; nie wszystkie z nich mogą być interpretowane przez automat podobnie jak przez człowieka, np. obrazy czy animacje.

Nowe technologie wykorzystywane w tworzeniu stron internetowych
Wśród nich można wymienić:
– JavaScript[2],
– AJAX[3],
– Flash[4],
– Silverlight[5].

Nowe technologie, wspomagające prezentację treści oraz pracę z zasobami internetowymi, zmieniają sposób interakcji użytkownika ze stroną. Dzięki skryptom wykonywanym przez przeglądarkę witryny zyskują bardziej dynamiczny wygląd, a dzięki asynchronicznej komunikacji z serwerem zawartość strony może być szybko aktualizowana, dodatkowe oprogramowanie zaś (na przykład w postaci rozszerzeń przeglądarki) może zapewnić dynamiczny, jednoekranowy interfejs. Jednakże odejście od schematu wyłącznych żądań HTTP na rzecz częstszego wykonywania skryptów i korzystania przez użytkowników przeglądarek z zewnętrznych aplikacji, stawia kolejne wyzwanie systemom automatyzacji. Muszą one rozpoznawać

[2] Więcej na ten temat na stronie https://developer.mozilla.org/pl/JavaScript [dostęp: 30.12.2012].
[3] Więcej na ten temat na stronie http://www.adaptivepath.com/ideas/essays/archives/000385.php [dostęp: 30.12.2012].
[4] Więcej na ten temat na stronie http://www.adobe.com/pl/products/flash/ [dostęp: 30.12.2012].
[5] Więcej na ten temat na stronie http://www.silverlight.net/ [dostęp: 30.12.2012].

i wykorzystywać elementy strony dostępne jedynie przez wykonanie skryptu, muszą monitorować dynamiczne zmiany na stronie, w tym te wywołane przez AJAX. Pożądaną właściwością jest także obsługa elementów wykraczających poza strukturę obiektowego modelu dokumentu[6] (ang. *Document Object Model*, DOM), takich jak obiekty Flash.

2. Istniejące rozwiązania automatyzacji czynności użytkownika w przeglądarce

Z automatyzacją czynności w przeglądarce związanych jest wiele problemów, które nie mają prostego rozwiązania. Warto się przyjrzeć, w jaki sposób istniejące systemy radzą sobie ze wspomnianymi wyzwaniami.

2.1. Chickenfoot

Chickenfoot to rozszerzenie przeglądarki Mozilla Firefox, składające się z biblioteki języka JavaScript oraz edytora wbudowanego w przeglądarkę [Bolin 2005]. Skrypty Chickenfoot pisane są w języku JavaScript, dzięki czemu użytkownicy systemu otrzymują dostęp do pełnych możliwości tego języka, na przykład `window`, `document`, także z możliwością modyfikacji zawartości strony przy użyciu obiektowego modelu dokumentu.

Jednakże skrypty Chickenfoot nie przestrzegają modelu bezpieczeństwa dla języka JavaScript i wykonywane są z pozycji uprzywilejowanej, dzięki czemu mają dostęp do całości przeglądarki, wszystkich odwiedzanych stron, a także do systemu plików użytkownika oraz sieci. Rozszerza to możliwości skryptów w obszarze integracji informacji z różnych źródeł, a jednocześnie może prowadzić do naruszenia bezpieczeństwa tychże informacji.

Cechą wyróżniającą system Chickenfoot jest intuicyjne działanie wykorzystujące odnajdowanie wzorców [Little i Miller 2006]. Aby móc operować na danym elemencie strony, większość poleceń przyjmuje za argument pewną charakterystykę tego elementu, dzięki czemu można go jednoznacznie zidentyfikować, stosując odpowiedni algorytm wyszukiwania wzorców. Podstawą do wykonania algorytmu są słowa kluczowe lub opis relatywnego położenia elementu.

Chickenfoot zdolny jest odtworzyć wiele standardowych akcji podejmowanych przez użytkownika, jak kliknięcia, praca z formularzami czy nawigacja między stronami. Dodatkowo istnieje możliwość wykonania akcji dostępnych jedynie z poziomu programu, w tym dostosowanie wyglądu strony, a także pobranie strony w formie obiektu JavaScript, na przykład w celu wstępnej analizy.

Jednocześnie tworzenie i wykonywanie skryptów pozostaje w dobrze znanym użytkownikowi środowisku – w przeglądarce internetowej, gdzie nieprzerwanie

[6] Więcej na ten temat na stronie http://www.w3.org/DOM/ [dostęp: 30.12.2012].

można kontrolować wyniki przeprowadzonych operacji, bez konieczności analizowania i rozumienia kodu HTML. Cel, jaki przyświeca Chickenfoot, można wyrazić następująco: użytkownik nie powinien być zmuszony do oglądania kodu źródłowego strony w celu jej dostosowania i automatyzacji [Bolin i Miller 2005].

2.2. Koala

Koala to system umożliwiający nagrywanie i automatyzację procesów biznesowych w sieci, a także późniejsze współdzielenie opisu tych procesów. Rozwiązanie jest dostarczane jako dodatek do przeglądarki Mozzilla Firefox.

Koala wykorzystuje paradygmat programowania przez demonstrację [Rubin 1986], dzięki czemu akcje użytkownika mogą być przechwytywane, edytowane i odtwarzane w formie poleceń pseudonaturalnego języka programowania. Polecenia tego języka mogą być łatwo zrozumiane i edytowane przez człowieka. Elastyczność systemowego interpretera poleceń jest na tyle duża, że jako skrypty udaje się z powodzeniem wykorzystywać instrukcje napisane oryginalnie dla ludzi. Dzięki temu praca z kodem staje się niezwykle intuicyjna.

Zastosowanie języka programowania w takiej postaci było możliwe dzięki podejmowaniu każdej akcji w kontekście konkretnej strony – z uwzględnieniem kolekcji dostępnych elementów oraz akcji. Przyjmując polecenia w formie zbliżonej do naturalnych zdań, na przykład „wpisz Danny w pole imię", system przeprowadza kilka wnioskowań:

– określa kolekcję etykiet dla każdego elementu strony,
– dopasowuje słowa kluczowe w poleceniu (na przykład: „wpisz", „pole") do zawartości strony,
– pozyskuje argumenty wykonywanych akcji (jak „Danny") poprzez analizę składni polecenia.

Następnie, po określeniu największego prawdopodobieństwa dla różnych wariantów interpretacji, wykonuje operacje.

Skrypty systemu Koala domyślnie są zapisywane na stronach Wiki, zwanych *Koalescence*, dzięki czemu opis wykonywanych operacji może być współdzielony. Dodatkowo, przy wykorzystaniu korporacyjnych i indywidualnych źródeł informacji, możliwe jest automatyczne dostosowywanie działania udostępnionych skryptów do konkretnych użytkowników.

Koala pozwala kolekcjonować i współdzielić informacje o czynnościach wykonywanych przez użytkownika w przeglądarce, w ramach wykonywania danego etapu w procesie biznesowym, przez co wspomaga interakcje człowieka z systemem oraz usprawnia przepływy pracy [Little i in. 2007].

2.3. Greasemonkey

Greasemonkey, podobnie jak prezentowane wcześniej systemy Chickenfoot i Koala, jest rozszerzeniem przeglądarki Mozilla Firefox. Dzięki niemu użytkownicy

mogą dodawać do dowolnej strony internetowej własne skrypty (ang. *user scripts*) napisane w języku JavaScript. Skrypty te mogą dostosowywać niemal każdy aspekt zachowania strony – jej wygląd oraz sposoby interakcji – umożliwiając dynamiczne zmiany kodu HTML [McFarlane 2005]. Kod użytkownika jest uruchamiany przy każdej wizycie na stronie, a zmiany, jakich dokonują dodatkowe skrypty, są bezpośrednio włączane do dokumentu HTML, bez pozostawiania oznak swojej obecności. Greasemonkey może być narzędziem do wzbogacania stron internetowych o nowe funkcjonalności, zmiany wyglądu prezentowanej zwartości, łączenia danych z wielu źródeł oraz do wielu innych celów.

Chociaż skrypty uruchamiane przez Greasemonkey mogą być bardzo użyteczne, to tworzenie ich wymaga znajomości języka JavaScript oraz budowy strony, na której są uruchamiane. Skrypty użytkownika nie mogą wprawdzie uszkodzić danych przechowywanych na dysku, ale błędnie napisany kod może w znaczącym stopniu wpłynąć na pracę przeglądarki, a jeżeli jest napisany w złej wierze – może się stać programem szpiegującym aktywność użytkownika w sieci [Pilgrim 2005].

2.4. Smart Bookmarks

Inteligentne zakładki (ang. *Smart Bookmarks*) to system zbudowany jako pasek boczny dla przeglądarki Mozilla Firefox. Posługując się nim, internauci mogą zachowywać odnośniki odwiedzanych stron w formie zakładek, powracać do zapisanych wcześniej stron oraz edytować graficzną reprezentację zakładek. System nieustannie analizuje i rejestruje czynności użytkownika i – działając w tle – zachowuje dane o tych czynnościach. Odmiennie od tradycyjnej historii odwiedzanych stron, zachowywane są nie tylko adresy URL, ale również czynności związane z akcjami wskaźnika myszy na stronie oraz wprowadzaniem danych do formularzy.

W momencie zapisania strony w formie zakładki system określa sekwencję zdarzeń, jakie muszą być wygenerowane, aby odtworzyć bieżący stan strony, a następnie zapisuje tę sekwencję. Istotną kwestią jest określenie momentu, od którego wykonywane czynności wpływały na ostateczny wygląd i stan zapisywanej strony. Zakładka jest odtwarzana poprzez wykonywanie przyporządkowanych do niej poleceń w odpowiedniej sekwencji.

Elementem, który wyróżnia system *Smart Bookmarks* spośród innych programów automatyzujących pracę z zasobami internetowymi, jest podejście do przechwytywania sekwencji czynności (zwanych makrami) automatycznie i działając wstecz, bez konieczności włączania nagrywania. Pozwala to na definiowanie zadań automatyzacji w czasie pracy ze stronami internetowymi, jak też dobrze wpisuje się w znany użytkownikom model zachowywania odnośników do stron w formie zakładek.

Dodatkowo, system *Smart Bookmarks* oferuje szeroką wizualizację wykonywanych działań, zawierającą wyjaśnienia tekstowe, zdjęcia ekranu oraz animacje. Tekstowa reprezentacja zakładek jest zbudowana z użyciem jedynie słów kluczowych,

Tabela 2. Porównanie funkcjonalności systemów automatyzacji czynności użytkownika w przeglądarce

Cecha	Chickenfoot	Koala	Greasemonkey	Smart Bookmarks
Dodatek do przeglądarki	+	+	+	+
Skrypty użytkownika	+	+	+	–
Wykorzystanie JavaScript w skryptach	+	–	+	–
Pseudonaturalny język skryptowy	–	+	–	–
Rejestrowanie czynności użytkownika	–	+	–	+
Możliwość współdzielenia	+	+	+	+

dzięki czemu użytkownik może edytować wykonywany kod bez konieczności poznawania składni języka skryptowego [Hupp i Miller 2007].

Zaprezentowane systemy pozwalają na automatyzację czynności wykonywanych przez użytkownika zarówno przez ręcznie zdefiniowane skrypty, jak i przez odtwarzanie zarejestrowanych zachowań użytkowników przeglądarki. Porównanie funkcjonalności tych systemów prezentuje tabela 2. Przedstawione rozwiązania są związane głównie z przeglądarką Mozilla Firefox, co jest spowodowane jej dużą popularnością wśród internautów, a także otwartą architekturą, pozwalającą na tworzenie dodatków rozszerzających jej funkcjonalności.

3. Możliwości zastosowania automatyzacji w realizacji procesów biznesowych

Ewolucja aplikacji internetowych w kierunku rozbudowanych systemów informacyjnych przynosi nowe wymagania w stosunku do tychże aplikacji, między innymi konieczność zarządzania złożonymi procesami wykonywanymi przez wielu użytkowników w różnych organizacjach, poprzez łączenie różnorodnego oprogramowania [Brambilla i in. 2006]. W odpowiedzi na potrzeby implementacji procesów biznesowych w aplikacjach internetowych, wiele modeli i metodyk, wykorzystywanych przy projektowaniu informacyjnych aplikacji internetowych, zostało rozbudowanych o możliwości modelowania takich procesów. Wśród nich należy wymienić: Object Oriented Hypermedia Design Method (OOHDM), Web Site Design Method (WSDM), Object Oriented Hypermedia (OO-H) i metodologię UWE, Web Modelling Language (WebML) oraz framework Ubiquitous Web Applications (UWA) [Distante, Rossi i Canfora 2007].

Uogólniając, można wyróżnić dwa główne paradygmaty projektowania uwzględniającego implementację procesów biznesowych w aplikacjach internetowych [Rossi i Turrini 2008]:

– rozwijanie modelu nawigacyjnego w celu odwzorowania przepływów pracy,
– rozwijanie notacji modelowania procesów w celu dostarczania użytkownikowi sekwencji widoków w trakcie interakcji z aplikacją internetową.

Wykorzystanie automatyzacji czynności podejmowanych przez użytkownika przeglądarki internetowej wspomaga procesy biznesowe na innym poziomie. Zamiast rozwijania możliwości projektowania nowych systemów łączących przepływy pomiędzy różnymi procesami, automatyzacja pozwala na zastąpienie pracy człowieka, będącego uczestnikiem wielu procesów, i integrację działań istniejących aplikacji.

Automatyzacja działań w przeglądarce pozwala wspomagać przede wszystkim wykonywanie pojedynczych etapów zdefiniowanych procesów biznesowych, wymagających interakcji użytkownika z aplikacją internetową. Możliwe jest wyeliminowanie powtarzających się sekwencji akcji, wykonywanych przez niektórych aktorów procesu biznesowego – poprzez zastąpienie ich uruchamianym skryptem. Dzięki temu praca ludzi może się skupiać na elementach procesów, których nie można zautomatyzować, w tym na wykonywaniu zadań twórczych lub podejmowaniu decyzji eksperckich. Przykładami zastosowań rozwiązań automatyzacji czynności w przeglądarce mogą być:

– automatyczne logowanie do serwisu lub aplikacji internetowej, na przykład do panelu administracyjnego systemu kontroli zawartości strony internetowej,
– automatyczne tworzenie zapytania do systemu wyszukiwawczego poprzez wypełnianie formularzy oraz ustalanie wartości pól wyboru i opcji, na przykład podczas wyszukiwania najnowszych wiadomości na zadany temat w serwisie informacyjnym,
– automatyczna analiza i przetworzenie wyników wyszukiwania, w tym zastosowanie niestandardowego filtrowania i sortowania, formatowanie wyników, rozszerzanie lub ograniczenie uzyskiwanej informacji, na przykład podczas analizowania ofert sprzedaży w sklepach internetowych,
– wykonanie sortowania elementów, według zadanego algorytmu, przy wykorzystaniu interfejsu typu *drag&drop*, na przykład podczas ręcznego przydzielenia zasobów do projektów,
– prezentowanie określonej sekwencji stron internetowych.

Wykorzystanie technik automatyzacji w powyższych przykładach wiąże się z koniecznością określenia pożądanych sekwencji akcji w taki sposób, aby system automatyzujący mógł je jednoznacznie wykonać. Jednakże dzięki wykorzystaniu automatyzacji czynności użytkownika w przeglądarce, oprócz zmniejszenia ilości wykonywanej rutynowo pracy, możliwe jest osiągnięcie wielu celów, między innymi:

– zmniejszenie zawodności pracy człowieka – zmniejszone zostaje ryzyko błędu powstałego wskutek pomyłki człowieka, na przykład nieprawidłowego wprowadzenia wartości lub pominięcia jakiegoś działania,
– uszczegółowienie wykonywanych procesów biznesowych – opis procesów biznesowych zbudowany jest zwykle na pewnym poziomie abstrakcji, pozostawia uczestnikowi procesu pewną dowolność działań; w celu automatycznego wykonywania pewnych zadań, kolejne kroki muszą zostać szczegółowo opisane,
– eksternalizacja wiedzy ukrytej – poprzez przedstawienie schematu działań w postaci wykonywalnej przez system automatyzujący następuje ujawnienie części specjalistycznej wiedzy o danym procesie, zdobytej poprzez praktykę,
– tworzenie procesów ad hoc – dodatkowe procesy mogą być tworzone poprzez wyspecyfikowanie pożądanych sekwencji akcji,
– rozproszona kompozycja procesu – dzięki łączeniu indywidualnych opisów części procesów, zbudowanych przez ich uczestników, może nastąpić kompozycja procesu w sposób rozproszony i niezależny.

Zastosowanie technik automatyzacji w przeglądarce internetowej przy wspomaganiu procesów biznesowych ma swoje ujemne strony. Można wśród nich wymienić:
– konieczność włożenia pewnego wysiłku w utworzenie opisu procesu dla systemu automatyzującego,
– możliwość powstania niespójności w procesie komponowanym niezależnie,
– trudność w obsłudze sytuacji niestandardowych, wymagających decyzji eksperta.

Mimo tych niedoskonałości, automatyzacja w przeglądarkach jest warta rozważenia, szczególnie w środowiskach, w których spoiwem pomiędzy procesami i działaniami w odrębnych systemach jest praca ludzka wykonywana w aplikacjach internetowych.

4. Możliwe podejścia do budowy zestawu poleceń automatyzacji wspomagającego procesy biznesowe

Można rozważać kilka poziomów, na jakich mogą być wykonywane działania zastępujące pracę użytkownika, począwszy od podstawowych operacji w warstwie protokołów sieciowych, aż do próby bezpośredniej interpretacji przez system opisu procesu w języku naturalnym. Ocena pozytywnych i negatywnych konsekwencji wyboru konkretnego podejścia stała się podstawą do sformułowania i implementacji prezentowanego w artykule zestawu poleceń automatyzacji czynności użytkownika w przeglądarce internetowej.

Zaczynając od działań na najniższym poziomie, jako podstawę do automatyzacji procesów w przeglądarce można zaproponować działania w warstwie protokołów sieciowych. Jest to wariant zakładający w dużym stopniu odseparowanie wykonywanych czynności od graficznego interfejsu przeglądarki, w tym wizualnej

reprezentacji strony. Na przykład, zamiast wpisania loginu i hasła do pól w formularzu i zatwierdzenia tego formularza poprzez kliknięcie przycisku, do serwera wysyłane jest odpowiednie polecenie POST protokołu HTTP, zawierające konkretne dane. W proces komunikacji włączony jest jedynie system automatyzacji określający i wysyłający żądania w ramach protokołu sieciowego oraz serwer, który reaguje w określony sposób.

Przykładem programu, który może być pomocny w tworzeniu takiego rozwiązania, jest cURL[7], będący narzędziem wiersza poleceń. Ma on rozbudowane możliwości zarządzania informacjami przesyłanymi przez poszczególne protokoły.

Wśród zalet automatyzacji działającej na poziomie protokołów sieciowych należy wymienić:

- znaczną kontrolę nad wykonywanymi działaniami – komunikacja z serwerem może być sterowana poprzez określenie postaci i sposobu wysyłania kolejnych żądań,
- niewielkie zużycie zasobów komputera – dzięki oddzieleniu wykonywanych operacji od graficznego interfejsu użytkownika,
- szybkość wykonywania.

Rozwiązanie wykorzystujące bezpośrednią obsługę protokołów sieciowych wiąże się również z wieloma niedogodnościami [Müffke 2001], wśród których można wymienić:

- obsługę wymagającą wiedzy eksperckiej – należy posiadać stosowną wiedzę o wymianie danych w sieci i używaniu protokołów sieciowych,
- konieczność analizy kodu źródłowego strony – wysyłanie żądań do serwera wymaga znajomości szczegółowej budowy strony internetowej w celu prawidłowego naśladowania jej zachowania,
- trudność w obsłudze niektórych elementów graficznych – działanie w pewnym oderwaniu od graficznej reprezentacji strony utrudnia wykonywanie zadań automatyzacji, między innymi poprzez ukrywanie zależności między obiektami oraz utrudnianie przewidywania rezultatów działań.

Odmiennym podejściem do budowania zestawu poleceń automatyzacji może być wykorzystanie języków skryptowych. Pozwala ono zarówno na obsługę graficznych elementów strony dostępnych dla użytkownika, jak i na zastosowanie programowania do wykonywania zaawansowanych operacji. Taka metoda daje szeroki wachlarz możliwości zarządzania zdarzeniami na stronach internetowych, pozwalając na budowanie złożonych rozwiązań automatyzacji i integracji [Ahmadi i in. 2009].

W takiej konwencji został utworzony język Chickenfoot, a także, do pewnego stopnia, Greasymonkey. Występuje tu możliwość naśladowania akcji użytkownika na stronie internetowej, takich jak lokalizacja dostępnych elementów, kliknięcie

[7] Więcej na ten temat na stronie http://curl.haxx.se/ [dostęp: 30.12.2012].

czy wybranie pozycji z listy, ale także możliwość uruchomienia dowolnego kodu JavaScript.

Zastosowanie języków skryptowych jako podstawy dla poleceń automatyzacji ma wiele pozytywnych aspektów, między innymi:

- szerokie możliwości operacyjne – korzystanie z rozbudowanego instrumentarium języków skryptowych ułatwia budowanie złożonych struktur na potrzeby automatyzacji,
- elastyczność rozwiązania – dostosowanie do indywidualnych potrzeb jest ułatwione poprzez rozszerzenie podstawowych możliwości pracy ze stroną,
- rozbudowana kontrola nad stroną – w dużej mierze dzięki działaniu w obrębie obiektowego modelu dokumentu,
- efektywność użytkowania – budowanie nowych rozwiązań staje się wydajniejsze za sprawą szerokich możliwości kompozycji poleceń wykorzystywanego języka.

Do wad takiego podejścia można zaliczyć:

- trudność w tworzeniu sekwencji automatyzacji – wymagana jest pewna znajomość programowania skryptowego,
- połączenie funkcjonalności dostępnych dla użytkownika przeglądarki oraz funkcjonalności dostępnych jedynie przez wykonanie skryptu – wprowadza to pewną niespójność w projektowaniu systemów automatyzujących pracę człowieka.

Systemy automatyzacji wykorzystujące języki programowania w przeglądarce adresują wiele istotnych wymagań dla tego typu rozwiązań i mogą stanowić solidny fundament dla budowania narzędzi wspomagających procesy biznesowe.

Zastosowanie programowania „niestarannego" (ang. *sloppy programming*) może stanowić podstawę do budowania systemów automatyzacji przeglądarki, ze szczególnym uwzględnieniem łatwości użytkowania. Przykładem takiego systemu jest Koala, pozwalający niemal w sposób naturalny formułować ciągi akcji wykonywanych na stronie internetowej.

Dzięki takiej budowie rozwiązania można osiągnąć między innymi:

- przyjazność dla użytkownika – system nie wymaga posiadania szczególnej wiedzy specjalistycznej, a wywoływanie działań może przyjmować formę zbliżoną do języka naturalnego,
- możliwość wykorzystywania instrukcji napisanych dla ludzi, w tym przenoszenia opisów procesów biznesowych wprost do systemu,
- popularyzację rozwiązania, szczególnie wśród osób bez znajomości technicznych aspektów programowania.

Zwiększenie przyjazności użytkowania systemu powoduje zwykle występowanie liczniejszych problemów w innych obszarach jego działania, na przykład:

- podatności na błędy – niejednoznaczność opisów akcji w języku naturalnym może powodować działanie odmienne od planowanego,
- większego zużycia zasobów – interpretacja poleceń nieposiadających określonej struktury wymaga głębszej analizy, w tym syntaktycznej, dla prawidłowego wykonywania programu.

Systemy wykorzystujące programowanie „niestaranne" mają znaczący potencjał, jednak ich rozwój jest uwarunkowany możliwościami przetwarzania języka naturalnego.

5. Proponowane rozwiązanie

W badaniach prowadzonych przez autora przyjęto rozwiązanie nieco odmienne od prezentowanych podejść. Wykorzystując ustrukturyzowaną formę poleceń automatyzacji, zbiór dostępnych akcji dostosowano do możliwości, jakie posiada użytkownik przeglądarki. Skuteczny system automatyzacji czynności użytkownika, mający wspomagać procesy biznesowe, powinien mieć wyraźnie nakreślone ramy działania. Kiedy celem jest odwzorowanie akcji podejmowanych przez uczestnika procesu w trakcie odwiedzania kolejnych stron, istotne wydaje się wyspecyfikowanie zakresu funkcjonalności odpowiadającego tym konkretnym potrzebom. Dzięki obsłudze jedynie wyselekcjonowanych, intuicyjnych dla użytkownika zdarzeń, zestaw poleceń staje się wyspecjalizowanym narzędziem do odtwarzania sekwencji akcji podjętych przez użytkownika oraz automatycznego naśladowania zachowania internautów [Rees 2002].

Analiza dostępnych rozwiązań oraz możliwości spełnienia postawionych wymagań, poprzez rozwój istniejącego oprogramowania, doprowadziły do określenia koncepcji implementacji zestawu poleceń automatyzacji, który opiera się na trzech filarach. Pierwszym z nich jest Interfejs Programowania Aplikacji, do omawianego wcześniej języka Chickenfoot, który stał się podstawą dla prezentowanego rozwiązania. Odrzucone zostały metody, które nie odpowiadają czynnościom wykonywanym standardowo przez użytkownika w przeglądarce, w tym te dotyczące zmiany wyglądu strony. Nowe metody w zestawie poleceń, obejmujące między innymi obsługę kliknięć określonego punktu obiektu na stronie, zostały wprowadzone w celu obsługi zewnętrznych technologii wspierających, jak na przykład Flash lub Silverlight. W tabeli 3 zaprezentowano przygotowany zestaw poleceń automatyzacji.

Drugim filarem rozwiązania jest komponent programistyczny o nazwie csEXWB, który udostępnia pełną funkcjonalność przeglądarki internetowej. Dzięki niemu możliwe jest wykonywanie poleceń, działając na graficznej reprezentacji strony internetowej. Jego główne zadanie to tworzenie, przechwytywanie i obsługa zdarzeń związanych z oryginalnym komponentem przeglądarki. Otwarta budowa komponentu csEXWB [csEXWB Project 2010] pozwoliła na rozszerzenie go o interfejs zbudowany z wykorzystaniem API języka Chickenfoot. Wskutek tego możliwe stało się bezpośrednie zarządzanie zachowaniem przeglądarki za pomocą zestawu poleceń automatyzacji.

Wreszcie trzecim elementem, stanowiącym podstawę skutecznego działania omawianej implementacji, jest aplikacja Deep Web Data Integration (http://integrator.net/en/content/view/17/65/, dostęp: 30.12.2012), rozwijana przez projekt Integror

Tabela 3. Polecenia automatyzacji czynności użytkownika w przeglądarce

Nazwa polecenia automatyzacji	Opis
CheckBox	zaznaczenie pola wyboru
CheckRadio	zaznaczenie pola opcji
ClickButton	kliknięcie przycisku
ClickHTMLElement	kliknięcie dowolnego elementu HTML
ClickLink	kliknięcie hiperłącza
Click	kliknięcie określonego punktu dowolnego elementu strony internetowej za pomocą wywołania systemowego
DragDrop	wykonanie przeciągnięcia i upuszczenia dowolnego elementu HTML na dowolny inny element, z określeniem punktu początkowego i końcowego, za pomocą wywołania systemowego
EnterTextToTextArea	wprowadzenie tekstu do wieloliniowego pola tekstowego
EnterTextToTextBox	wprowadzenie tekstu do pola tekstowego
EnterText	wprowadzenie tekstu w dowolnym elemencie HTML poprzez symulację wprowadzania znaków z klawiatury
NavigateBack	nawigacja wstecz
NavigateForward	nawigacja wprzód
NavigateToUrl	nawigacja do określonego adresu URL
Pick	zaznaczenie określonego elementu na liście wyboru
UncheckBox	usunięcie zaznaczenia z pola wyboru
UncheckRadio	usunięcie zaznaczenia z pola opcji
Unpick	usunięcie zaznaczenia z określonego elementu na liście wyboru

(http://integrator.net/en/, dostęp: 30.12.2012), która wykorzystuje możliwości języka Chickenfoot, włączając go w swoją strukturę. Celem aplikacji jest wykorzystanie wydajnego formalizmu do ekstrakcji ustrukturyzowanych dokumentów ze źródeł tak zwanego głębokiego Internetu [Calì i Martinenghi 2010]. Stosując zaczerpniętą z teorii automatów reprezentację rozbudowanych źródeł danych w Internecie oraz zasady ekstrakcji danych za pomocą relatywnego XPath [Berglund i in. 2010], aplikacja pozwala na powtarzalne pozyskiwanie dokumentów za pomocą zdefiniowanych lub częściowo zdefiniowanych zapytań.

Wśród zalet podejścia, ograniczonego do funkcjonalności interfejsu przeglądarki, można wymienić:

– spójność zestawu poleceń – występują w nim jedynie funkcje odnoszące się do akcji, jakie może podjąć użytkownik,

– specjalizację – na podstawie takiego zestawu poleceń tworzone są rozwiązania w celu automatycznego wykonywania czynności podejmowanych przez użytkownika, bez rozwijania pobocznych ścieżek zarządzania stroną, jej wyglądem i zachowaniem,
– łatwość w odzwierciedleniu rzeczywistych procesów – rozwiązania budowane na podstawie zestawu poleceń tego typu mogą lepiej oddać przebieg rzeczywistych akcji, które podejmują użytkownicy.

Wadami tego spojrzenia na budowę zestawu poleceń automatyzacji mogą być:
– ograniczona funkcjonalność – brak funkcji związanych z pracą ze stroną poza interfejsem graficznym może być wadą dla niektórych rozwiązań,
– mniejsza elastyczność – poprzez zwiększenie specjalizacji zmniejsza się elastyczność w wykorzystywaniu zestawu poleceń do szerokiego spektrum zastosowań.

Proponowane rozwiązanie nie uwzględnia wszystkich wyzwań, jakie stoją przed systemami zastępującymi pracę ludzką w przeglądarce, jednak dzięki połączeniu kilku technologii możliwości, jakie prezentuje ta aplikacja, pozwalają na skuteczne zastosowanie jej jako katalizatora dla procesów biznesowych.

Podsumowanie

Automatyzacja czynności użytkownika w przeglądarkach internetowych i wykorzystanie jej we wspomaganiu procesów biznesowych to ciągle rozwijające się dziedziny, obejmujące z jednej strony element ludzki, czyli użytkownika i uczestnika procesu, z drugiej zaś – techniczne aspekty przekazywania zdarzeń i wprowadzania danych do mechanizmu przeglądarki. Obserwowany jest nieustanny postęp na tym polu, zwłaszcza w aspekcie nowych sposobów interakcji pomiędzy użytkownikiem a aplikacją internetową, a także w obszarze wykorzystywanych technologii wspomagających przetwarzanie informacji. Jak przedstawiono w niniejszym artykule, odpowiedni dobór mechanizmów automatyzujących działania w przeglądarce pozwala wspierać definiowanie oraz realizację procesów biznesowych, eliminując powtarzalne czynności wykonywane przez użytkownika.

Bibliografia

Abramowicz, W., 2008, *Filtrowanie informacji*, Wydawnictwo Akademii Ekonomicznej w Poznaniu, Poznań.

Adar, E., Teevan, J., Dumais, S.T., 2009, *Resonance on the Web: Web Dynamics and Revisitation Patterns*, w: *CHI '09: Proceedings of the 27th International Conference on Human Factors in Computing Systems*, ACM, New York, s. 1381–1390.

Ahmadi, N., Jazayeri, M., Lelli, F., Repenning, A., 2009, *Towards the Web of Applications: Incorporating End User Programming into the Web 2.0 Communities*, w: *SoSEA'09:*

Proceedings of the 2nd International Workshop on Social Software Engineering and Applications, ACM, New York, s. 9–14.

Anupam, V., Freire, J., Kumar, B., Lieuwen, D., 2000, *Automating Web Navigation with the WebVCR*, w: *Proceedings of the 9th International World Wide Web Conference on Computer Networks: The International Journal of Computer and Telecommunications Netowrking*, North-Holland Publishing Co., Amsterdam, The Netherlands, s. 503–517.

Atterer, R., Wnuk, M., Schmidt, A., 2006, *Knowing the User's Every Move: User Activity Tracking for Website Usability Evaluation and Implicit Interaction*, w: *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, ACM, New York, s. 203–212.

Berendt, B., Spiliopoulou, M., 2000, *Analysis of Navigation Behaviour in Web Sites Integrating Multiple Information Systems*, The VLDB Journal, vol. 9, no. 1, s. 56–75.

Berglund, A., Boag, S., Chamberlin, D., Fernández, M.F., Kay, M., Robie, J., Siméon, J., 2010, *XML Path Language (XPath) 2.0 (Second Edition)*, http://www.w3.org/TR/xpath20/ [dostęp: 20.11.2012].

Bolin, M., 2005, *End-User Programming for the Web*, praca magisterska, Massachusetts Institute of Technology.

Bolin, M., Miller, R.C., 2005, *Naming Page Elements in End-User Web Automation*, w: *WEUSE I: Proceedings of the First Workshop on End-User Software Engineering*, ACM, New York, s. 1–5.

Bolin, M., Webber, M., Rha, P., Wilson, T., Miller, R.C., 2005, *Automation and Customization of Rendered Web Pages*, w: *UIST '05: Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, ACM, New York, s. 163–172.

Brambilla, M., Ceri, S., Fraternali, P., Manolescu, I., 2006, *Process Modeling in Web Applications*, ACM Transaction on Software Engineering Methodology, vol. 15, no. 4, s. 360–409.

Calì, A., Martinenghi, D., 2010, *Querying the Deep Web*, w: *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, ACM, New York, s. 724–727, http://doi.acm.org/10.1145/1739041.1739138 [dostęp: 20.11.2012].

Catledge, L.D., Pitkow, J.E., 1995, *Characterizing Browsing Strategies in the World-Wide Web*, Computer Networks and ISDN Systems, vol. 27, no. 6, s. 1065–1073.

csEXWB Project, 2010, http://code.google.com/p/csexwb2/ [dostęp: 30.12.2012].

Distante, D., Rossi, G., Canfora, G., 2007, *Modeling Business Processes in Web Applications: An Analysis Framework*, w: *SAC '07: Proceedings of the 2007 ACM Symposium on Applied Computing*, ACM, New York, s. 1677–1682.

Faaborg, A., Lieberman, H., 2006, *A Goal-Oriented Web Browser*, w: *CHI '06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, New York, s. 751–760.

Goecks, J., Shavlik, J., 2000, *Learning Users' Interests by Unobtrusively Observing their Normal Behavior*, w: *IUI '00: Proceedings of the 5th International Conference on Intelligent User Interfaces*, ACM, New York, s. 129–132.

Hong, J.I., Landay, J.A., 2001, *WebQuilt: A Framework for Capturing and Visualizing the Web Experience*, w: *Proceeding of the 10th International Conference on World Wide Web*, Hong Kong, s. 717–724.

Hupp, D., Miller, R.C., 2007, *Smart Bookmarks: Automatic Retroactive Macro Recording on the Web*, w: *UIST '07: Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, ACM, New York, s. 81–90.

Kellar, M., Watters, C., Shepherd, M., 2006, *The Impact of Task on the Usage of Web Browser Navigation Mechanisms*, w: *GI '06: Proceedings of Graphics Interface 2006*, Canadian Information Processing Society, Toronto, Canada, s. 235–242.

Little, G., Lau, T.A., Cypher, A., Lin, J., Haber, E.M., Kandogan, E., 2007, *Koala: Capture, Share, Automate, Personalize Business Processes on the Web*, w: *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, New York, s. 943–946.

Little, G., Miller, R.C., 2006, *Translating Keyword Commands into Executable Code*, w: *UIST '06: Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, ACM, New York, s. 135–144.

McFarlane, N., 2005, *Fixing Web Sites with Greasemonkey*, Linux Journal, vol. 2005, no. 138, s. 1.

Milic-Frayling, N., Jones, R., Rodden, K., Smyth, G., Blackwell, A.F. i Sommerer, R., 2004, *Smartback: Supporting Users in Back Navigation*, w: *Proceedings of the 13th International World Wide Web Conference*, New York, USA, s. 63–71.

Mueller, F., Lockerd, A., 2001, *Cheese: Tracking Mouse Movement Activity on Websites. A Tool for User Modeling*, w: *CHI '01: CHI '01 Extended Abstracts on Human Factors in Computing Systems*, ACM, New York, s. 279–280.

Müffke, F., 2001, *The Curl Programming Environment*, Dr. Dobb's Journal, vol. 26, no. 9, s. 66–71.

Pilgrim, M., 2005, *Dive Into Greasemonkey*, Mark Pilgrim, http://web.archive.org/web/20110726001221/http://diveintogreasemonkey.org/ [dostęp: 20.11.2012].

Reeder, R.W., Pirolli, P., Card, S.K., 2001, *WebEyeMapper and WebLogger: Tools for Analyzing Eye Tracking Data Collected in Web-Use Studies*, w: *CHI '01: CHI '01 Extended Abstracts on Human Factors in Computing Systems*, ACM, New York, s. 19–20.

Rees, M.J., 2002, *Evolving the Browser Towards a Standard User Interface Architecture*, w: *AUIC'02: Proceedings of the Third Australasian Conference on User Interfaces*, Australian Computer Society, Inc., Darlinghurst, Australia, s. 1–7.

Rossi, D., Turrini, E., 2008, *Designing and Architecting Process-aware Web Applications with EPML*, w: *SAC '08: Proceedings of the 2008 ACM Symposium on Applied Computing*, ACM, New York, s. 2409–2414.

Rubin, R.V., 1986, *Language Constructs for Programming by Example*, w: *COCS'86: Proceedings of the Third ACM-SIGOIS Conference on Office Information Systems*, ACM, New York, s. 92–103.

Safonov, A., Konstan, J.A., Carlis, J.V., 2001, *End-user Web Automation: Challenges, Experiences, Recommendations*, w: Lawrence-Fowler W.A., Hasebrook J. (eds.), *WebNet*, AACE, s. 1077–1085.

Tauscher, L., Greenberg, S., 1997, *Revisitation Patterns in World Wide Web Navigation*, w: *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, ACM, New York, s. 399–406.

WEB BROWSER AUTOMATION FOR BUSINESS PROCESSES SUPPORT

Abstract: Automation of user activities in a Web browser allows for streamlining of Internet applications usage, especially with respect to the repetitive tasks performed by Internet users. Thus, the possibility of automation systems can be used in the execution of specific elements of business processes that require interaction with Web applications. This paper presents an analysis of opportunities to automate repetitive tasks of Web browser users in the field of business process execution. The analysis of currently available solutions for browser automation allowed for evaluation of the usefulness of this class of applications for supporting business processes. Moreover, a set of commands was proposed to perform browser automation that takes into account the standard set of Internet users' activities.