POLITECHNIKA GDAŃSKA
Wydział Elektroniki, Telekomunikacji i Informatyki

Magdalena Godlewska

Model otwartej architektury rozproszonych dokumentów elektronicznych wspierającej proces podejmowania decyzji w trybie obliczeń zespołowych

Rozprawa doktorska

Promotor:
prof. dr hab. inż. Bogdan Wiszniewski
Wydział Elektroniki, Telekomunikacji i Informatyki
Politechnika Gdańska

Gdańsk, 2013

Spis treści

Wstęp

Dokument pełni kluczową rolę w strukturze każdej organizacji ludzkiej, będąc dla człowieka utrwaloną w kulturze jednostką informacji, a także jednostką interfejsu. W szczególności, zjawisko to można zaobserwować w organizacjach opartych na wiedzy, których celem jest podejmowanie decyzji, odkrywanie faktów czy gromadzenie wiedzy, np. w sądownictwie, pracy zespołów dochodzeniowych, zespołowej pracy badawczej, medycynie, itp. Całkowita automatyzacja procesów realizowanych w organizacjach tego typu, w oparciu o systemy informatyczne, nie jest możliwa, ze względu na kluczową rolę, jaką w ich procesach odgrywają ludzie, zobowiązani do rozstrzygania problemów, dla których rozwiązania algorytmiczne najczęściej nie istnieją. Systemy informatyczne dają możliwość wsparcia procesów podejmowanych przez organizacje ludzkie poprzez interakcję z człowiekiem i automatyzację części zadań związanych z realizacją procesu.

Rozprawa podejmuje zagadnienia związane z rozproszonym przetwarzaniem zespołowym (ang. *collaborative computing*) i pracą grupową (ang. *virtual collaboration*) z wykorzystaniem metod i narzędzi inżynierii dokumentu – nowej dyscypliny informatyki zapoczątkowanej na Uniwersytecie Berkeley w 2001 roku przez Roberta Glushko i Tima McGratha. Istotą tej dyscypliny jest poszukiwanie architektur i sposobów reprezentacji dokumentów elektronicznych niezależnych od systemów informatycznych wspierających ich tworzenie, interpretację, obieg i przechowywanie. Dzięki odpowiedniej strukturze i wbudowanej funkcjonalności, dokumenty mogą być czytelne równocześnie dla człowieka i komputera.

Proponowana w rozprawie architektura rozproszonych, mobilnych i interaktywnych dokumentów MIND wspiera realizację procesów podejmowanych przez organizacje ludzkie w dwóch aspektach: *komunikacji treści* i *koordynacji czynności*. Wsparcie komunikacji treści jest realizowane poprzez prawidłowe jej odtworzenie na różnych urządzeniach lokalnych użytkowników oraz wsparcie pracy z zawartością dokumentów za pośrednictwem różnego rodzaju usług: automatycznych bądź umożliwiających interakcję z użytkownikiem. Koordynacja czynności procesowych zapewnia zaś prawidłowy przepływ dokumentów pomiędzy uczestnikami pracy grupowej i wykonywanie na nich zadań określonych w procesie organizacyjnym.

Wszystkie kluczowe informacje związane z procesem są wbudowane w dokument, zatem model przetwarzania zespołowego, właściwy dla tego podejścia, określa się mianem przetwarzania skoncentrowanego na dokumencie (ang. *document-centric*) i przeciwstawia klasycznemu podejściu skoncentrowanemu na danych, oferowanemu przez inżynierię systemów i baz danych. Podejście skoncentrowane na dokumencie pozwala implementować rozwiązania o charakterze „lekkim" (ang. *lightweight*) charakteryzujące się znaczną trwałością i stabilnością, w odróżnieniu od „ciężkich" rozwiązań systemowych, dość szybko starzejących się wraz z technologiami zastosowanymi do ich realizacji.

Inżynieria dokumentu została uznana przez Narodowe Centrum Nauki za dyscyplinę leżącą w zakresie badań podstawowych, dzięki czemu badania nad proponowaną w rozprawie architekturą mogły być częściowo realizowane w ramach projektu badawczego MEtody i NArzędzia Inżynierii Dokumentu przyszłości (MENAID)[1]. W ramach tego projektu, oprócz architektury mobilnych dokumentów interaktywnych MIND opisanej w tej rozprawie, rozwijana jest także koncepcja dokumentów o wykonywalnej treści w oparciu o ideę interaktywnego dokumentu warstwowego oraz model procesu automatycznych negocjacji w przypadku konfliktów, które mogą wystąpić przy przetwarzaniu treści dokumentów w ramach pracy grupowej.

[1]Projekt badawczy nr 2011/01/B/ST6/06500, strona internetowa: http://menaid.org.pl/

Rozdział 1

Dokument elektroniczny

Dokument, najogólniej, to nośnik zrozumiałej dla człowieka treści, zwanej potocznie informacją. Stanowi naturalny, utrwalony przez wieki interfejs dla człowieka oraz powszechny środek komunikacji międzyludzkiej. Jako dokument mogą być rozumiane wszelkie środki przekazu informacji, od tradycyjnych dokumentów tekstowo-obrazkowych, odbieranych za pośrednictwem wzroku, po coraz bardziej powszechne dokumenty multimedialne, szczególnie dźwiękowe, ale też odbierane poprzez dotyk (np. napisane alfabetem Braille'a) bądź nawet przez węch[1].

Powszechny dostęp człowieka do urządzeń cyfrowych oraz sieci Internet przyczynia się nieustannie do rozwoju i wzrostu znaczenia dokumentów w postaci cyfrowej. Główną zaletą dokumentów cyfrowych jest łatwość i szybkość przekazywania ich kolejnym osobom, co umożliwia zdalną pracę grupową. Paradoksalnie, różnorodność notacji dokumentów często uniemożliwia prawidłowy odczyt zawartości dokumentu po przeniesieniu go na inne urządzenie cyfrowe.

Rozdział ten dokonuje przeglądu różnych notacji dokumentów, w celu ich klasyfikacji pod kątem komunikacji między zdalnymi użytkownikami. Przegląd uwzględnia rozwój modeli dokumentów, jaki nastąpił na przestrzeni ostatnich kilku dekad: od statycznej postaci nośnika informacji do dokumentów mobilnych i aktywnych. Na zakończenie proponuje nowy model mobilnego i interaktywnego dokumentu (MIND), który stanowi główny rezultat moich badań prezentowanych w rozprawie. Model ten umożliwia i wspiera pracę grupową w rozproszonych organizacjach ludzkich, w ramach tzw. *organizacji wirtualnych*. Charakteryzują się one fizycznym rozproszeniem współpracowników, ale wykorzystują mechanizmy komunikacji stwarzające iluzję pracy w jednym miejscu, w pewnym zdefiniowanym przedziale czasu i w ramach pewnego wspólnie realizowanego przedsięwzięcia.

[1] Popularny był kiedyś zwyczaj perfumowania listów, co pozwalało identyfikować jego nadawcę.

1.1. Notacje reprezentacji informacji

Format dokumentu, jako ustalony standard zapisu informacji definiujący jego strukturę i zawartość, jest jedną z głównych cech rozróżniających dokumenty [Chr06]. Istnieje wiele przyczyn występowania różnych formatów. Najczęściej są to względy komercyjne, tj. wspieranie pewnych formatów przez firmy informatyczne lub uzależnienie rodzaju zapisu danych od określonego oprogramowania dostępnego na rynku. Często w wyborze formatu istotną rolę pełni jego łatwość użycia, siła wyrazu, uniwersalność, sposób kompresji oraz zrozumiałość dla użytkowników. Formaty można, w ogólności, podzielić na otwarte, gdy opis standardu jest powszechnie dostępny oraz zamknięte, gdy szczegóły standardu znane są tylko producentowi.

Typ dokumentu jest pojęciem niezwiązanym bezpośrednio z technicznym opisem struktury dokumentu. Jest to pewna nazwa nadana konkretnym rodzajom dokumentów w celu powiązania ich z aplikacjami, które mogą je prawidłowo odczytać [Chr06]. Ze względu na to, że istnieje bardzo wiele typów dokumentów, ich właściwa identyfikacja jest kluczowa w przypadku konieczności przenoszenia dokumentów między różnymi urządzeniami. W takiej sytuacji pomocne są ogólnodostępne repozytoria, które zapewniają jednolity opis typów dokumentów dla wielu ich odbiorców.

1.1.1. Repozytorium typów dokumentów

Dominującym obecnie mechanizmem wymiany dokumentów różnych typów jest poczta elektroniczna. Dokument przesyłany jako wiadomość elektroniczna (email) może zawierać w sobie dokumenty dowolnego typu w postaci zarówno treści wiadomości jak i załączników. Mimo że wiadomość elektroniczna jest przekształcana na ciąg znaków możliwych do przesłania przez protokoły pocztowe [Kle08], agenty pocztowe są w stanie odebrać i odpowiednio odtworzyć zawartość wiadomości. Jest to możliwe dzięki standardowi MIME (ang. *Multipurpose Internet Mail Extensions*) [FB96], powszechnie stosowanemu przy przesyłaniu poczty elektronicznej.

Wiadomość w formacie MIME składa się z nagłówków, które określają atrybuty wiadomości i treści. Domyślne wartości tych atrybutów czynią je opcjonalnymi, co pozwala agentom pocztowym opartym na MIME również na właściwą interpretację wiadomości nie będących w tym formacie. Wiadomość w formacie MIME może składać się z wielu części (ang. *multipart*) różnych typów, z których każda część (ang. *part*) opisana jest przez nagłówek `Content-Type`, określający typ zawartości. Identyfikator typu podawany w nagłówku `Content-Type` składa się z typu głównego, podtypu oraz opcjonalnie z parametrów.

Przykładowo, identyfikator `text/html; charset=utf-8` określa, że zawartość

danej części wiadomości należy do typu głównego `text` zdefiniowanego dla zawartości tekstowych zrozumiałych dla człowieka i programów do ich automatycznego przetwarzania, podtypem jest `html`, czyli zawartość jest dokumentem w formacie HTML (ang. *Hypertext Markup Language*), parametr `charset` precyzuje sposób kodowania znaków dokumentu, czyli w tym przypadku `utf-8`. Identyfikator `application/msword` określa typ główny jako `application`, czyli typ specyficzny dla pewnej aplikacji, podtyp `msword` precyzuje, że jest to dokument programu Microsoft Word w formacie DOC (zgodnym z wersją tego narzędzia sprzed roku 2007).

Repozytorium typów i podtypów zawartości MIME jest tworzone i powszechnie udostępniane przez organizację Internet Assigned Numbers Authority (IANA) [FK05]. Repozytorium zawiera 9 typów głównych: `application`, `audio`, `example`, `image`, `message`, `model`, `multipart`, `text` i `video`. Każdy z tych typów zawiera wiele podtypów, a repozytorium daje możliwość rejestracji nowych. Nowe podtypy mogą:

- być standardowe, czyli ich standard musi być opisany w formalnie zatwierdzonej publikacji,
- należeć do konkretnych dostawców, ich nazwy rozpoczynają się wówczas od przedrostka `vnd.`
- znajdować się jeszcze w fazie eksperymentów czy być częścią produktu, który nie jest dystrybuowany komercyjnie, określone są wtedy jako prywatne (ang. *personal*) i rozpoczynają się od przedrostka `prs`.

Do dokumentów w formacie MIME można także dodawać zawartość typu niezarejestrowanego w IANA, co sprawia, że MIME daje możliwość przesłania dokumentu dowolnego typu. Typy lub podtypy znane, ale nie zarejestrowane w IANA, zapisywane są z przedrostkiem `x-`. Na przykład znana i standardowa zawartość programów napisanych w języku JAVA lub C otrzyma „niestandardowy” identyfikator: `text/x-java` lub `text/x-csrc`. Typy niestandardowe, nie rozpoznane przez MIME oznaczane są zaś jako `application/octet-steam`.

Aktualnie typy zawartości MIME są wykorzystywane znacznie szerzej niż tylko do opisywania zawartości poczty elektronicznej. Mogą służyć ogólnie do opisywania zawartości dowolnych dokumentów przeznaczonych do pracy grupowej. Sposób opisu treści dokumentu zgodny ze standardem MIME został wykorzystany w systemie opracowanym w rozprawie.

1.1.2. Notacje dokumentów a praca grupowa

Z jednej strony postać cyfrowa ułatwia przekazywanie dokumentów, z drugiej różnorodność notacji ogranicza możliwości ich prawidłowego odczytu i edycji. Typy zawartości MIME wspierają pracę grupową opartą na dokumencie i pozwalają zidentyfikować typ niemal każdego dokumentu napisanego przez człowieka, co umożliwia jego przesyłanie z wykorzystaniem protokołów poczty elektronicznej oraz powiązanie go z odpowiednią aplikacją. Nie rozwiązuje to jednak dwóch problemów związanych z przekazywaniem dokumentów. Oba dotyczą właściwego wyboru aplikacji do odpowiedniego przetwarzania nietekstowych zawartości:

1. szczegóły odtworzenia otrzymanej zawartości opisanej znanym typem MIME mogą wymagać wcześniejszych uzgodnień z nadawcą wiadomości,

2. niestandardowa zawartość wiadomości może wymagać od odbiorcy komunikatu odnalezienia i zainstalowania odpowiedniej aplikacji.

Sytuacja 1. zazwyczaj nie ma miejsca, gdy praca nad dokumentem odbywa się w grupie, która posiada takie samo oprogramowanie do edycji zawartości danego typu (tego samego producenta i tej samej wersji). Jednak jest to sytuacja rzadka w realiach organizacji wirtualnych, których członkowie znajdują się w terytorialnym rozproszeniu, zmieniają swoje lokalizacje i urządzenia, na których pracują (wykonują, przykładowo, część pracy w biurze a część w domu).

Jako przykład można tu podać dosyć częstą sytuację współpracy w oparciu o dokument stworzony w programie Microsoft Word z rozszerzeniem DOC. Załóżmy, że Anna edytuje dokument DOC, korzystając z aplikacji Microsoft Word i po skończonej pracy przesyła dokument pocztą elektroniczną do Bartosza, który ma wprowadzić istotne zmiany i odesłać dokument z powrotem do Anny. Wiadomość od Anny w formacie MIME będzie miała identyfikator zawartości dokumentu ustalony jako `application/msword`, co wskazuje Bartoszowi, że aplikacją właściwą do odpowiedniego wyświetlenia zawartości jest Microsoft Word. Jednakże Bartosz znajduje się już w domu, gdzie posiada komputer z systemem Linux i jedyną aplikacją, którą posiada do edycji formatu DOC jest LibreOffice Writer, który nie jest w pełni kompatybilny z Microsoft Word – szczególnie przy formatowaniu tabel. Zatem Bartosz staje przed dylematem: czy edytować dokument, umieszczając konieczne informacje, ale prawdopodobnie zmieniając formatowanie (co przysporzy Annie dodatkowej pracy) czy poczekać z pracą do powrotu do biura, gdzie ma dostęp do w pełni kompatybilnej aplikacji. To wprowadzi jednak opóźnienie w pracy nad dokumentem. Bartosz, oczywiście, może skonsultować swoją decyzję z Anną, wiąże się to jednak z pewną liczbą dodatkowych komunikatów, które

będą musieli wymienić między sobą. Sam typ zawartości ustalony w nagłówku MIME nie wystarcza do automatyzacji decyzji podejmowanej przez Bartosza. Potrzebna jest analiza zawartości całego dokumentu, a nie tylko specyfikacja typu jego zawartości. Problem ten określamy jako *potencjalną niekompatybilność* znanej zawartości dokumentu na lokalnym systemie odbiorcy wiadomości.

Sam typ zawartości MIME może być przez agenta pocztowego określony w sposób niejednoznaczny, co może przyczynić się do błędnej lub niedokładnej interpretacji zawartości przez odbiorcę. Przykładowo dokument RTF (Rich Text Format), którego zawartości odpowiada typ `text/rtf` lub `application/rtf` może otrzymać typ `application/msword`, co sugeruje agentowi pocztowemu odbiorcy dokumentu wybór aplikacji Microsoft Word oraz rozszerzenia DOC. Przy określaniu typu zawartości załącznika wiadomości brane jest też pod uwagę rozszerzenie nazwy pliku a nie zawartość pliku. Tak więc plik w formacie RTF bez żadnego rozszerzenia będzie rozpoznany jako zawartość niestandardowa, czyli typ `application/octet-stream`.

Zawartość niestandardowa, czyli nie rozpoznana przez MIME (lub po prostu nieznana odbiorcy dokumentu) może powodować sytuację wymienioną w punkcie 2. (str. 4.). Odbiorca, który nigdy nie miał do czynienia z danym typem zawartości, może oczekiwać wskazówek od nadawcy, jakie narzędzie powinien użyć lub jaką aplikację zainstalować, aby prawidłowo przetwarzać otrzymaną zawartość.

Przykładem ilustrującym tę sytuację może być proces pisania artykułu na konferencję przez trzech współautorów: Annę, Bartosza i Celinę. Wymogiem organizatorów konferencji jest, aby dokument napisany był w formacie LaTeX. Załóżmy, że Anna jest biegłym użytkownikiem zarówno formatu LaTeX jak i systemu operacyjnego Linux. Bartosz jest chętnym, lecz początkującym użytkownikiem systemu Linux. Celina jest sceptycznie nastawiona do rozpoczęcia pracy z nowym systemem i preferuje korzystanie z systemu Windows. Inicjatorem dokumentu jest Anna, tworząc jego konspekt, dzieląc pracę i umieszczając wstępną wersję bibliografii. Gotowe pliki wysyła jako załączniki pocztą elektroniczną do współautorów. Są to pliki: TEX z tekstem artykułu, BIB z bibliografią oraz DVI z wygenerowanym dokumentem wyjściowym gotowym do obejrzenia. Klient pocztowy używający typów zarejestrowanych w IANA nada tym załącznikom identyfikator `application/octet-stream`. Może też nadać niestandardowe identyfikatory zawartości MIME dla tych załączników, kolejno: `text/x-tex`, `text/x-bibtex`, `application/x-dvi`. Bartosz po odebraniu wiadomości, chcąc rozpocząć pracę z dokumentem, zainstaluje pakiet `texlive-latex-base`. Jednak mimo to zauważy, że nie może dodać bibliografii oraz zobaczyć wynikowej wersji dokumentu. W tej sytuacji Anna podpowie mu, że powinien zainstalować dodatkowe pakiety: `biblatex` do obsługi bibliografii oraz `dvilx` umożliwiający przeglądanie dokumentów DVI.

Wymaga to dodatkowej komunikacji z Anną i poświęcenia czasu na dostosowanie systemu, który dla Bartosza jest nowym środowiskiem pracy. Celina również ma problem z odtworzeniem zawartości, co więcej Anna nie potrafi jej pomóc, gdyż nigdy nie instalowała narzędzi do obsługi dokumentów LaTeX na systemie Windows. Celina musi więc poświęcić czas na znalezienie odpowiednich aplikacji. Znajduje kompletny zestaw narzędzi do wszystkich plików związanych z technologią LaTeX, np. MiKTeX [Sch12]. W związku z niestandardowym typem zawartości dokumentu praca nad artykułem opóźni się z powodu konieczności dostosowania lokalnych systemów współautorów do jego edycji.

Powyższy przykład ilustruje problem niestandardowej zawartości dokumentów, która może wymagać dostosowania lokalnego systemu odbiorcy do jej przetwarzania. Konfiguracja lokalnego systemu leży po stronie użytkownika i wymaga od niego dodatkowych umiejętności lub konieczności konsultacji z innymi uczestnikami pracy grupowej. Problem ten można określić jako konieczność dostosowania lokalnego systemu przez odbiorcę dokumentu do *prawidłowego odtworzenia* nieznanego typu zawartości.

Identyfikacja typu zawartości MIME wyodrębnia i klasyfikuje składowe (załączniki) wiadomości, jednak nie gwarantuje, że odbiorca będzie potrafił wykonać przypisaną mu pracę na otrzymanej zawartości. Typy MIME bowiem opisują zawartość, nie analizując jej.

Jednym ze sposobów rozwiązania problemu potencjalnej niekompatybilności zawartości wspomnianego wcześniej jest grupowa edycja dokumentów w oparciu o model klient-serwer. Zarówno dokument jak i aplikacja do jego przetwarzania znajdują się wówczas na serwerze, użytkownicy pracy grupowej zaś posiadają tylko aplikację kliencką, umożliwiającą dostęp *on-line* do zawartości dokumentu. Użytkownik nie musi więc wiedzieć, jaka aplikacja jest używana do przetwarzania zawartości. Dokument nie jest również przesyłany drogą mailową, co jest szczególnie istotne w przypadku plików dużych rozmiarów. Idea ta przyczyniła się do powstania różnych pakietów biurowych dostępnych online (ang. *online office productivity suites*), opartych o technologię przechowywania danych w „chmurze" (ang. *cloud storage*) oraz umożliwiających dostęp do narzędzi i aplikacji za pośrednictwem przeglądarki internetowej niezależnie od posiadanego systemu operacyjnego. Najbardziej znanymi przykładami takich „chmur", które umożliwiają tworzenie, grupową edycję, przechowywanie i synchronizację dokumentów są Microsoft SkyDrive [Mic07] oraz Google Drive [Goo12]. Głównym założeniem tych aplikacji jest udostępnienie zasobów pamięci dyskowej w celu przechowywania plików oraz ich synchronizacji z plikami na lokalnym komputerze. Udostępniają one również aplikacje do tworzenia i edycji różnego typu dokumentów, m.in.: dokumentów tekstowych, prezentacji czy arkuszy

kalkulacyjnych. Są to jednak programy o okrojonych możliwościach w porównaniu z pakietami Microsoft Office czy Libre Office. Poza tym wymagają stałego dostępu do Internetu – w przypadku Google Drive w ogóle nie można kontynuować pracy przy zerwanym połączeniu, zaś w przypadku Microsoft SkyDrive można kontynuować pracę, tyle że po zerwaniu połączenia nie ma możliwości zapisu zmian. Istnieje również możliwość pobrania dokumentu na dysk lokalny w celu pracy *off-line*. Tworzone są dwie kopie dokumentu: na serwerze i na urządzeniu lokalnym. Potrzebna jest wówczas konwersja dokumentu znajdującego się na serwerze do formatu obsługiwanego przez aplikację użytkownika, a następnie synchronizacja lokalnej wersji dokumentu z powrotem na serwerze. Ta operacja może jednak przyczynić się do wystąpienia problemu potencjalnej niekompatybilności zawartości, podobnie jak w przypadku wysyłania dokumentów pocztą elektroniczną.

Przykładowo, w celu grupowej edycji, został utworzony dokument poprzez interfejs Google Drive. Ma on wówczas format GDOC, który nie jest obsługiwany przez aplikacje Microsoft Word czy LibreOffice Writer. Zawartością formatu GDOC jest w istocie adres URL oraz ID dokumentu znajdującego się na serwerze, więc nie ma możliwości edycji tego dokumentu przy użyciu żadnej lokalnej aplikacji. Wymagana jest stała łączność z serwerem lub skorzystanie z możliwości pobrania dokumentu przekonwertowanego do jednego z kilku dostępnych formatów, m.in. DOCX, ODT, RTF czy PDF. Załóżmy, że dokument został pobrany w formacie DOCX. Można go teraz edytować lokalnie, jednak jego zawartość nie zostanie zsynchronizowana z wersją GDOC. Nie ma też możliwości edycji dokumentu DOCX poprzez interfejs Google Drive – należy wpierw dokonać konwersji tego pliku na format GDOC. Tak więc operacja pobrania dokumentu, lokalnej edycji i ponownej edycji na serwerze tworzy aż trzy niezsynchronizowane kopie dokumentu oraz psuje formatowanie (najczęściej tabel) w dokumencie.

W aplikacji Microsoft SkyDrive został zastosowany inny mechanizm tworzenia i synchronizacji dokumentów. Dokument jest od razu tworzony w jednym z formatów z pakietu Microsoft Office: Word, Excel, PowerPoint lub OneNote. Tak więc zapisanie dokumentu na dysku lokalnym nie wymaga konwersji a dokument zostaje zsynchronizowany. Jednak okrojona funkcjonalność aplikacji dostępnej przez serwer uniemożliwia edycję sformatowanych tabel, czy też dodatkowych obiektów, np. wzorów matematycznych. Tak więc do tworzenia dokumentów często nie wystarcza aplikacja udostępniana przez serwer, zaś różne wersje aplikacji lokalnych mogą przyczynić się do wystąpienia opisanego wcześniej problemu potencjalnej niekompatybilności zawartości.

Grupowa edycja w oparciu o model klient-serwer pozwala na korzystanie z aplikacji udostępnionych *online* przez serwer. Jednak jeśli użytkownik chce mieć

dostęp do dokumentu *off-line* na swoim lokalnym urządzeniu, może to powodować również problem prawidłowego odtworzenia nieznanego typu zawartości (problem 2. na str. 4). Serwery mogą udostępniać aplikacje operujące na dedykowanych formatach, które są niestandardowe lub nieznane użytkownikowi. Bądź odwrotnie, użytkownik chce pracować na dokumencie danego formatu, ale żadna aplikacja wspierająca ten format nie jest dostępna na serwerze. Można tu przytoczyć choćby wcześniejszy przykład grupowego pisania artykułu w środowisku LaTeX. W takiej sytuacji autorzy mogą wykorzystywać jedną z opisanych wcześniej „chmur" jedynie do wymiany plików, zaś problem związany z dostosowaniem lokalnego środowiska do przetwarzania danej zawartości pozostaje nierozwiązany.

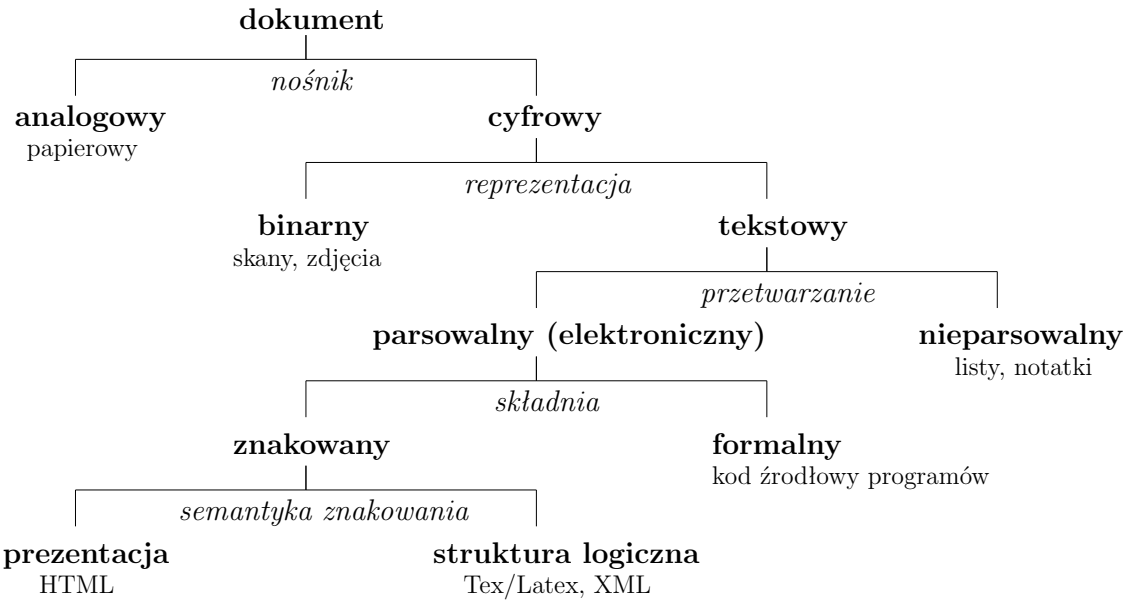
Reasumując, pierwszą wadą rozwiązania pracy grupowej z dokumentem w oparciu o model klient-serwer jest konieczność pozostawania online za każdym razem, gdy dokument jest przetwarzany. Może to być uciążliwe (połączenie może zanikać) oraz kosztowne, szczególnie w przypadku płatnego transferu danych przez sieć komórkową. Pobieranie dokumentów i praca *off-line* może zaś wymagać dostosowania lokalnego środowiska do pracy z daną zawartością, co może spowodować problemy wymienione wcześniej (str. 4). dotyczące wyboru odpowiedniej aplikacji. Drugą wadą jest narzucanie użytkownikom aplikacji nie zawsze zgodnych z ich preferencjami, ograniczonymi lub przestarzałymi. Użytkownicy, chcący napisać artykuł w Microsoft Word mogą wybrać rozwiązanie Microsoft SkyDrive, zaś chcąc napisać artykuł w LaTeX, mogą się zdecydować na ShareLaTeX [Sha13]. Wymaga to posiadania kilku kont, a więc loginów i haseł, a udostępniane przez serwer narzędzia mogą nie spełniać oczekiwań użytkowników. Mimo wkładu pracy użytkownika, często łatwiej i znacznie taniej jest dostosować urządzenie lokalne do przetwarzania danej zawartości niż dostosować serwer do przetwarzania nowej zawartości lub uaktualnianie istniejących aplikacji do obsługi zmieniających się formatów. Trzecią wadą z perspektywy prywatności i bezpieczeństwa jest umieszczanie treści dokumentów na zewnętrznych serwerach. Wydaje się oczywiste, że instytucje związane z sądownictwem, organami ścigania, czy inne organizacje gromadzące poufne bądź komercyjne dane powinny umieszczać dokumenty tylko na swoich wewnętrznych serwerach.

Opisane powyżej powszechne problemy w przetwarzaniu zawartości dokumentu nasuwają pytanie, czy i jak można by je wyeliminować, a przez to ułatwić grupową pracę w oparciu o dokumenty. Można by, na przykład, zastosować agenta użytkownika, który pośredniczyłby między dokumentem a współautorem, automatycznie analizując zawartości dokumentu i wspierając dostosowanie lokalnego systemu do odtwarzania jego zawartości. Trzeba jednak zauważyć, że typy dokumentów różnią się możliwościami analizy ich zawartości przez program komputerowy. Dlatego w następnym punkcie rozprawy dokonuję szczegółowej klasyfikacji typów dokumen-

tów ze względu na możliwość automatycznego przetwarzania ich zawartości.

1.1.3. Klasyfikacja typów dokumentów

Typy dokumentów można sklasyfikować w sposób przedstawiony na rysunku 1.1 [God06b]. Podział ten dotyczy możliwości przetwarzania zawartości dokumentów przez urządzenia elektroniczne. Ogólnie, segreguje dokumenty począwszy od zrozumiałych tylko przez ludzi do zrozumiałych jednocześnie przez ludzi i urządzenia elektroniczne[2].



Rys. 1.1. Klasyfikacja typów dokumentów

Dokumenty analogowe i cyfrowe

Dokumenty można podzielić ze względu na nośnik na dwie główne grupy: analogowe i cyfrowe. Jako *dokumenty analogowe* określone zostały wszystkie dokumenty będące w formie niezdygitalizowanej, czego przykładem są dokumenty papierowe (drukowane lub pisane odręcznie). Mogą to być także dowolne zasoby informacji, takie jak płyty winylowe, zielniki, albumy ze zdjęciami itp. obiekty informacyjne. Wiele organizacji, np. sądowniczych, wymaga obecności dokumentów analogowych ze względów formalnych, prawnych lub dlatego, że niosą one poza treścią dodatkowe informacje. Występuje również wiele historycznych dokumentów analogowych,

[2]Zrozumiałość, w tym przypadku, polega na możliwości automatycznej analizy treści przez program komputerowy.

istotnych dla różnych organizacji. Stąd zasadność umieszczenia dokumentów analogowych w klasyfikacji typów dokumentów.

Gdy dany dokument analogowy zostanie przetworzony na zrozumiałą dla urządzenia elektronicznego postać numeryczną, wówczas możemy mówić o *dokumencie cyfrowym*. Są także dokumenty cyfrowe typu „born electronic", które powstały w środowisku komputerowym od razu jako cyfrowe i nie mają swoich analogowych poprzedników.

Istnieje wiele formatów dokumentów cyfrowych, które mają różne możliwości i zastosowania. Zasadniczo można podzielić je odnośnie reprezentacji na dwie klasy: binarne i tekstowe, gdzie te pierwsze są nieprzetworzonym zapisem dokumentu analogowego w postaci zestawu próbek informacji, np. pikseli, zaś te drugie są pewnym zestawem rozpoznawalnych symboli, które mogą mieć różną złożoność i strukturę.

Dokumenty binarne

Dokumenty binarne będące przeniesieniem dokumentów analogowych (papierowych) do środowiska komputerowego są reprezentowane jako zestaw pikseli o określonych cechach. Przykładowe formaty takiego zapisu to JPG, BMP czy TIFF. Treść tego typu dokumentów jest w praktyce trudna do rozpoznania przez komputer ze względu na obecność szumów obniżających jego czytelność. Jednak taki dokument ma także zalety: może być przechowywany w pamięci komputera i przesyłany przez sieć, a po wyrenderowaniu na ekran lub wydruku na papierze może być przez człowieka traktowany jak oryginał. Nawet takie artefakty, jak plamy, zagniecenia czy tekstura papieru mogą nieść ważną informację dla jego użytkownika.

Ogólnie, analiza treści dokumentów w postaci binarnej wymaga wsparcia zewnętrznej aplikacji i jest związana z cyfrowym przetwarzaniem obrazów. Jest to problem złożony i wciąż otwarty, rozwiązywany na wiele sposobów metodami sztucznej inteligencji. Metody te są coraz bardziej skuteczne, jednakże nie pozbawione błędów – zwłaszcza w przypadku tekstów odręcznych lub mocno zaszumionych [CHSD12, BS11]. Dużo lepiej wygląda natomiast kwestia wizualizacji dokumentów binarnych i wiąże się po prostu z wyświetleniem pikseli. Oczywiście, pojawiają się tu problemy rozdzielczości, kompresji czy częstości próbkowania sygnału, ale są to kwestie obecnie dość dobrze rozwiązywane, ze względu na coraz lepsze możliwości sprzętu.

Dokumenty tekstowe

Dokumenty tekstowe zawierają wyłącznie ciągi symboli rozpoznawalne bezpośrednio przez komputer i człowieka. Ich podział uzależniony jest od możliwości przetwarzania zawartych w nich informacji. Można wyróżnić dokumenty parsowalne i nieparsowalne.

Nieparsowalne dokumenty tekstowe zawierają strumień znaków wyrażony w języku naturalnym, np. bezpośredni zapis rozmowy telefonicznej w formie stenogramu lub tekstu. Taka zawartość dokumentu jest oznaczona przez typ MIME: `text/plain`. Jedynym sposobem automatycznego wydobywania potrzebnych informacji z takiego dokumentu jest użycie narzędzi do przetwarzaniu języka naturalnego. Wizualizacja zaś przebiega na zasadzie kopiowania i wyświetlania strumienia tekstowego i jest całkowicie zależna od sposobu prezentowania go przez odpowiedni program do tego służący.

Grupą dokumentów znacznie łatwiej przetwarzanych przez komputer są dokumenty *parsowalne*, zwane również *dokumentami elektronicznymi*. Są to dokumenty posiadające oprócz treści, także strukturę, która jest zrozumiała zarówno przez człowieka jak i przez komputer. W celu zdefiniowania dokumentu elektronicznego można więc użyć następującej równości:

$$\textit{dokument elektroniczny} = \textit{struktura} + \textit{zawartość}.$$

Ze względu na składnię, dokumenty elektroniczne można podzielić na: znakowane i formalne. *Formalne* dokumenty elektroniczne zawierają kod źródłowy programów komputerowych, a ich gramatyka i składnia są ściśle sprecyzowane. W wiadomościach MIME są identyfikowane przez odpowiednie podtypy, przykładowo `text/x-java` dla parsowalnego formalnego kodu źródłowego w języku Java. Wydobywanie treści odbywa się poprzez rozbiór dokumentu przez odpowiednie analizatory składni (parsery) i jest w pełni skuteczne (np. narzędzie do generowania dokumentacji Javadoc [Ora10]). Dobrze zdefiniowana gramatyka określa jednoznacznie elementy składni: zmienne, deklaracje typów, instrukcje itp. Wyświetlanie dokumentów formalnych odbywa się również przez analizę składni i jest dokonywane przez narzędzia służące do prezentacji kodu źródłowego, np. edytory posiadające wsparcie dla danego języka i umożliwiające automatyczne formatowanie wydruku (ang. *pretty printing*).

Dokumenty *znakowane* zawierają w swojej treści fragmenty opisane odpowiednimi znacznikami. Znaczniki te organizują zawartość dokumentu, służąc jego prezentacji lub nadając mu strukturę logiczną. Gdy znaczniki wprowadzane są w celu zarządzania wyglądem dokumentu, tak jak w dokumentach HTML, wówczas ich wizualizacja jest bardzo prosta i przebiega „w locie” (jest interpretowana w trakcie

czytania dokumentu przez odpowiednią aplikację). Wydobywanie treści zaś odbywa się poprzez transformację – jest trudniejsze i mniej jednoznaczne niż w przypadku dokumentów posiadających strukturę logiczną, jak np. XML. W dokumentach ze strukturą logiczną znaczniki wprowadzane są w celu opisania zawartości, a nie wyglądu dokumentu, toteż wydobywanie treści odbywa się poprzez wyszukanie odpowiednich znaczników, co jest jednoznaczne i natychmiastowe. Wizualizacja wymaga zaś transformacji dokumentu do oczekiwanego formatu, np. w XML żądany wygląd można uzyskać za pomocą tzw. obiektów formatujących (ang. *formatting object*), np. XSL-FO [AMRZ01].

Zaprezentowana klasyfikacja typów dokumentów ukazuje różnorodność notacji informacji, a co za tym idzie złożoność problemu automatycznej analizy zawartości. Użytkownik w swojej pracy spotyka się z wieloma typami dokumentów, co nierzadko sprawia mu problemy i prowadzi do przedłużania się pracy z danym dokumentem. Zmienność istniejących formatów i pojawianie się nowych rodzi problemy związane z prawidłową identyfikacją i przetwarzaniem zawartości. Dlatego rozwój dokumentów powinien iść w kierunku ich kompatybilności oraz wsparcia użytkowników w pracy grupowej z różnorodną zawartością.

1.2. Rozwój dokumentów elektronicznych

Powszechność wymienionych wcześniej problemów potencjalnej niekompatybilności znanej zawartości lub prawidłowego odtworzenia nieznanej zawartości nasuwa pytanie, czy jest możliwe stworzenie agenta zdolnego do analizy treści dokumentu w celu wsparcia użytkownika w pracy nad tym dokumentem. Rozwój dokumentów elektronicznych potwierdza potrzebę dostosowania ich struktury wewnętrznej do analizy zawartości i wspierania pracy grupowej. Użytkownicy oczekują od dokumentów już nie tylko możliwości wygodnego przeglądania treści, ale też tego, by dokument zachowywał wszystkie zalety dokumentów w formie papierowej oraz oferował dodatkowe funkcje, w tym szczególnie wspierał pracę grupową, aktywnie zabezpieczał dostęp użytkowników do określonych fragmentów dokumentu, personalizował sposób wykorzystania zawartości, np. poprzez wyświetlanie jej w języku użytkownika, dawał możliwości nanoszenia przypisów, wprowadzania odsyłaczy, itd. Pojawiające się rozwiązania dają możliwość łączenia różnorodnych typów dokumentów w ramach jednej struktury oraz zmieniają rolę dokumentów w procesie ich przetwarzania ze statycznej w aktywną.

Rozwój dokumentów przyczynił się do powstania dziedziny informatyki zwanej *inżynierią dokumentu* (ang. *document engineering*) [GM05, GM08]. Jej głównym założeniem jest analiza i tworzenie użytecznych modeli i systemów generacji,

reprezentacji, obiegu i zarządzania informacją w oparciu o dokumenty, które mogą być dowolnego typu i formatu. Wprowadza ona paradygmat modelowania i implementowania systemów *skoncentrowanych na dokumencie* (ang. *document-centric*) zamiast klasycznego podejścia skoncentrowanego na danych (ang. *data-centric*). Podejście skoncentrowane na danych zakłada używanie identycznych struktur danych do wymiany informacji podczas interakcji współpracujących stron. Oznacza to, że dokumenty pełnią statyczną rolę w wymianie informacji, czyli są jedynie jednostką informacji. Podejście skoncentrowane na dokumencie zakłada, że dokument jest jednocześnie jednostką informacji i interfejsu, przez co może automatycznie uruchamiać odpowiednie aplikacje lub wspomagać użytkownika w pracy nad daną zawartością.

Nurt rozwoju dokumentów zmienia ich rolę w interakcji z użytkownikiem z pasywnej na aktywną. Dokument staje się aktywny, gdy oprócz zawartości i struktury posiada także *funkcjonalność*, czyli zbiór funkcji opisujących zachowanie dokumentu zależne np. od lokalizacji, wykonanych czynności czy profilu użytkownika. Dokumenty aktywne można podzielić na *reaktywne*, czyli reagujące na pewne zdarzenia oraz *proaktywne*, a więc samodzielnie generujące pewne zdarzenia lub inicjujące procesy. Zarówno dokumenty pasywne, reaktywne i proaktywne są obecnie powszechnie używane. Jako przykłady takich dokumentów można wymienić:

- pasywny dokument napisany w języku TeX, zawierający treść i strukturę,
- reaktywny dokument DOC z dodanymi makrami,
- proaktywny formularz ankiety badania rynkowego czy reklama pojawiające się samodzielnie na stronie internetowej.

W kolejnych podrozdziałach prezentuję historię rozwoju dokumentów z uwzględnieniem ich funkcjonalności, aby na koniec zaproponować własny model mobilnego i interaktywnego dokumentu MIND i sformułować tezę rozprawy.

1.2.1. Warstwowe dokumenty wielopostaciowe

Jednym z pomysłów pozwalających na łączenie różnych typów zawartości w ramach jednej struktury oraz zarządzania poszczególnymi częściami dokumentu jest model dokumentu warstwowego zaproponowany przez Phelpsa i Wilensky'ego w [PW96, PW98] i stanowiący punkt wyjściowy dla moich badań nad rozwojem dokumentów elektronicznych [God06a, God06b]. Pozwala on stworzyć jeden dokument składający się z kilku warstw o różnej zawartości. Warstwy, dzięki odpowiednim znacznikom, tworzą zazwyczaj strukturę drzewa – można je dowolnie ukrywać i uwidaczniać,

czyniąc ich treść opcjonalną. Taka struktura umożliwia nanoszenie przypisów, podkreśleń czy adnotacji, podobnie jak w dokumentach papierowych. Umieszczane są one w innych warstwach niż oryginalny dokument bazowy, co pozwala chronić jego treść a także ustalać politykę dostępu do naniesionych treści. Innym przykładem zastosowania warstw jest możliwość „nałożenia" tłumaczenia na istniejący już dokument zamiast tworzyć nowy dokument w innym języku. Warstwy umożliwiają również bardziej zaawansowane współdzielenie dokumentów. Użytkownicy mogą ustalać prawa dostępu do poszczególnych części dokumentu, np. zabezpieczać je hasłem. Pozwalają także na tworzenie dokumentów rozproszonych, którego poszczególne warstwy znajdują się na różnych urządzeniach elektronicznych i są wyświetlane jako całość przez odpowiednio do tego przystosowaną aplikację.

Format PDF (ang. *Portable Document Format*) jest pierwszym formatem dokumentów, w którym warstwowość została wprowadzona jako obowiązujący standard i jest rozwijana na dużą skalę. Począwszy od wersji PDF 1.5 [Ado04] został wprowadzony mechanizm opcjonalnej zawartości (ang. *optional content*) i dotyczy sekcji zawartości dokumentu, która może być dowolnie uwidaczniana i ukrywana przez autora dokumentu lub jego użytkownika. Sprawiło to, iż warstwy dokumentu stały się jeszcze bardziej praktyczne, gdyż istnieje możliwość dowolnej ich prezentacji, zarządzania pojedynczymi warstwami, podwarstwami a także grupami warstw. Opcjonalna zawartość pozwala na tworzenie warstw treści, rysunków, a także wielojęzycznych dokumentów. Zawartość warstw jest łączona w grupy OCG (ang. *optional content group*), które są podstawową strukturą używaną do sterowania widocznością składowych dokumentu. Reprezentują one pewne kolekcje elementów, które mogą być dynamicznie ukrywane bądź uwidaczniane przez użytkownika. Elementy należące do danej grupy mogą się znajdować w różnych miejscach dokumentu i należeć do różnych strumieni zawartości. Grupie przypisywany jest stan ON (włączona) lub OFF (wyłączona). W podstawowym przypadku zawartość należąca do jednej grupy jest widoczna, gdy stan jest ON i niewidoczna, gdy jest OFF, ale w bardziej złożonych przypadkach zawartość może należeć do kilku grup o wykluczających się stanach. Wówczas można ustalić pewną taktykę widoczności, dla zawartości należącej do konkretnego zgrupowania danych. Przykładowo:

```
<</Type /OCMD                    %grupowanie danych w OCMD
  /OCGs[12 0 R 13 0 R 14 0 R]    %słownik składa się z trzech grup
                                 %opcjonalnej zawartości
  /P /AllOn                      %stan P zostaje ustalony na AllOn, co
>>                               %oznacza, że zawartość jest widoczna,
                                 %jeśli stany każdej z trzech grup są ON,
                                 %w przeciwnym przypadku jest ukryta
```

Phelps i Wilensky [PW96, PW98] rozszerzają model warstwowy dodatkowo o funkcjonalność, proponując model *dokumentu wielopostaciowego* (MVD, ang. *multivalent document*). Dokument składa się z rozproszonej zawartości i dynamicznych programów, zwanych odpowiednio *warstwami* (ang. *layers*) i *zachowaniami* (ang. *behaviors*). Warstwy, jak opisano wyżej, dają możliwość włączenia różnych typów zawartości w strukturę jednego dokumentu. Mogą być też dodawane później do utworzonego już dokumentu. Warstwy odzwierciedlają semantykę zawartości, i nie wynikają z lokalizacji w dokumencie – można powiedzieć, są to *warstwy semantyczne*.

Funkcjonalność dokumentu MVD została zrealizowana przez Phelpsa i Wilensky'ego za pomocą dynamicznych obiektów wykonywalnych, zwanych zachowaniami – pozwalają one na aktywne manipulowanie zawartością dokumentu. Zachowania mogą komunikować się z trzema rodzajami komponentów dokumentu w dowolnej konfiguracji: z *danymi* tworzącymi zawartość warstw, z *interfejsem własnym* oraz innych zachowań, a także z *interfejsem użytkownika*. Zachowania mogą być aktywowane w dowolnej chwili czasu i dla różnych części dokumentu, użytkownik może dodawać także nowe zachowania. Przykładami wykorzystania zachowań są: wyszukiwanie i zaznaczanie informacji w dokumencie, dodawanie różnorodnych, spersonalizowanych i rozproszonych adnotacji czy zmiana wyświetlania zawartości, np. sortowanie tabel, tłumaczenie stron.

Architektura dokumentów wielopostaciowych MVD stanowi pierwszy rozproszony model dokumentów elektronicznych znany w literaturze inżynierii dokumentu. Zarówno warstwy jak i zachowania mogą znajdować się w terytorialnym rozproszeniu na różnych urządzeniach – lokalnych bądź serwerach połączonych siecią Internet i komunikujących się poprzez usługę WWW (World Wide Web). Model MVD został zrealizowany w oparciu o technologię Java, informacje o warstwach i zachowaniach są zebrane w dokumencie – rdzeniu (ang. *hub-document*), będącym dokumentem XML [Phe05].

Model MVD posiada zalety rozproszonego modelu warstwowego oraz rozszerzalną funkcjonalność pozwalającą manipulować treścią dokumentu, w tym zamieszczać różnorodne adnotacje. Model nie uwzględnia jednak możliwości przemieszczania się dokumentów między różnymi lokalizacjami, czy problemu utraty połączenia internetowego. Wprowadzona funkcjonalność pozwala dynamicznie uaktywnić i sterować samą zawartością dokumentu, jednak bez możliwości zarządzania dokumentami na wyższym poziomie w oparciu o interakcję z użytkownikiem.

1.2.2. Dokumenty aktywne

Dodanie funkcjonalności do dokumentu zmienia jego rolę w procesie przetwarzania z pasywnej na aktywną. Możliwość wykorzystania dowolnego języka programowania do specyfikacji zachowań dokumentu, czyni jego funkcjonalność otwartą i potencjalnie nieograniczoną. Stąd można zdefiniować *dokument aktywny* jako dokument elektroniczny zawierający wbudowany w niego zbiór usług, definiujący jego funkcjonalność [CTZ02], zatem:

$$\textit{dokument aktywny} = \textit{struktura} + \textit{zawartość} + \textit{funkcjonalność}$$

Pierwsza koncepcja w pełni aktywnych dokumentów elektronicznych zdolnych do samodzielnego działania niezależnie od aktualnej lokalizacji (ang. *placeless documents*) została zaproponowana przez Dourish i in. [DEL+00, Dou03]. Model ten wykorzystuje możliwość wbudowywania funkcjonalności w dokument w celu zarządzania jego zawartością. Głównym założeniem autorów było odejście od standardowej hierarchicznej struktury i ścisłej kategoryzacji dokumentów w katalogach systemów operacyjnych, systemach mailowych czy na stronach internetowych. Tradycyjnie bowiem umieszcza się dokument w konkretnym katalogu znajdującym się w pewnej statycznej hierarchii katalogów. Odszukanie takiego dokumentu może być utrudnione, zwłaszcza gdy zapisywany był zgodnie z innym kryterium niż później odszukiwany lub jeśli ma być odszukany przez inną osobę, która zupełnie tego kryterium nie znała. Autorzy zauważyli, że katalogi plików nie tylko zapewniają organizację dokumentów (grupowanie, lokalizację), ale także służą do zarządzania nimi, np. tworzenia archiwów czy umożliwiania zdalnego dostępu. Aby skuteczniej zarządzać dokumentami zaproponowali więc odejście od tradycyjnej hierarchii katalogów poprzez wprowadzenie własności (ang. *properties*) dokumentów, które pozwolą uniezależniać dokumenty od ich lokalizacji i dostosowywać je dynamicznie do kontekstu użycia.

Własności opisane są metadanymi zawierającymi cechy dokumentu zrozumiałe dla człowieka. Mogą mieć charakter publiczny, dostępny dla każdego użytkownika, jak i indywidualny, dostosowany do konkretnego odbiorcy. Użytkownicy sami mogą również dodawać i modyfikować własności zdefiniowane przez siebie. Własności nadawane dokumentom mogą być *statyczne* lub *aktywne* – to właśnie te drugie czynią dokument aktywnym. Aktywne własności oprócz nazwy i wartości zawierają bowiem kod programu, wprowadzający do dokumentów funkcjonalność i pozwalający na zarządzanie dokumentem. Własności mogą być wykorzystane do grupowania dokumentów, kontroli ich statusu i prezentacji, wyszukiwania, mogą także dostosowywać dokument do użytkownika czy pośredniczyć między aplikacjami, zarządzać wersjami i kopiami dokumentów.

Jednym ze sposobów zarządzania dokumentami w oparciu o aktywne własności jest praca grupowa z dokumentem (ang. *document-centered collaboration*), związana z przepływem pracy (ang. *workflow*), którego koordynowanie należy do dokumentu a nie do aplikacji [LED+99, Dou03]. Dzięki temu, dokument może sam sterować swoim stanem w procesie.

Implementacja modelu *placeless documents* rozróżnia dwa rodzaje dokumentów: *dokument bazowy*, zawierający treść oraz *dokument referencyjny*, zawierający wskaźnik do dokumentu bazowego. Oba rodzaje dokumentów mogą mieć dołączone własności, z tym że dokument bazowy może mieć tylko własności uniwersalne. Można także wyróżnić trzy rodzaje własności, różniące się chwilami ich wywołania przez system: weryfikator (ang. *verifier*) po przywołaniu pewnej operacji sprawdza czy ma ona dostęp do dokumentu, wykonawca (ang. *performer*) jest wykonywany w trakcie operacji, np. przekształca treść dokumentu oraz powiadamiający (ang. *notifier*) jest wykonywany po operacji, np. rejestruje zmiany.

System *Placeless documents* nie tyle *przechowuje* treść dokumentu, co ją *dostarcza*. Został zaimplementowany w języku Java i składa się z rozproszonych kontenerów, nazwanych przez twórców systemu jądrami (ang. *kernels*) – są to centralne jednostki zarządzające operacjami. Jednostki te gromadzone są w obrębie jednej przestrzeni (ang. *space*) i to właśnie przestrzeń identyfikuje dokumenty, choć fizycznie znajdują się one w obrębie jądra. Każde urządzenie musi mieć zainstalowaną aplikację jądra i przyłączyć się do jednej przestrzeni, by móc korzystać z *Placeless documents*. Własności dokumentów gromadzone są w relacyjnej bazie danych i identyfikowane przez niezmienne nazwy.

Placeless documents to rozwiązanie skoncentrowane na dokumencie, który jest aktywny oraz pośredniczy między użytkownikami i aplikacjami. Głównym założeniem tego rozwiązania jest wsparcie indywidualnej pracy z dokumentem i współdzielenie zasobów, choć możliwa jest też realizacja przepływu pracy. System wymaga jednak, aby na każdym urządzeniu była zainstalowana dosyć rozbudowana aplikacja jądra, która wymaga częstej łączności z internetem, co czyni go mało praktycznym dla małych urządzeń mobilnych, takich jak smartfony. Zabezpieczenie spójności dokumentu w razie utraty połączenia, jak i zabezpieczenie dostępu do dokumentu zostały zrealizowane poprzez obiekt pośredniczący (ang. *proxy object*) po stronie serwera, który również wymaga łączności internetowej. Ograniczeniem jest również fakt, że jedno jądro może należeć do jednej przestrzeni, co uniemożliwia jednoczesną pracę w wielu procesach, które mogą wymagać różnych przestrzeni.

1.2.3. Dokumenty jako agenty

Własności „wykonawcze" dokumentów *Placeless documents* są uruchamiane po zainicjowaniu akcji na dokumencie traktowanym jako obiekt. Takie dokumenty można nazwać *dokumentami reaktywnymi* (ang. *reactive documents*), gdyż reagują samodzielnie na każdą próbę dostępu do nich. Alternatywą są *dokumenty proaktywne* (ang. *proactive documents*), które posiadają pewną autonomię i same potrafią się uaktywnić lub dostosować do środowiska, w którym się aktualnie znajdują – mają cechy autonomicznego programu komputerowego zwanego agentem (ang. *software agent*). Tą klasę dokumentów będę nazywać *dokumentami-agentami* (ang. *document agents*), by podkreślić ich dwojaką naturę: klasycznie rozumianych dokumentów elektronicznych oraz agentów [CTZ02].

W [Sat01] została przedstawiona koncepcja dokumentów w postaci mobilnych agentów. Autor przedstawia implementację dwóch struktur: *MobileSpaces* i *MobiDoc*. *MobileSpaces* – to dedykowany system agentowy pozwalający na wykonywanie operacji i migrację agentów. Definiuje przy tym: hierarchię agentów, która oznacza, że agent może zawierać inne agenty, a także grupy migracji, które sprawiają, że agenty migrują do nowej lokalizacji jako całość, wraz z agentami, które są w nim zawarte. *MobiDoc* jest rozwiązaniem pozwalającym na tworzenie dokumentów składających się z różnego typu zawartości, wbudowanego kodu programu i zbioru serwisów (zaimplementowanych w języku Java) oraz na ustalanie protokołów komunikacji między komponentami dokumentu. Pozwala również na zarządzanie przepływem pracy, także w oparciu o pocztę elektroniczną.

System agentowy składa się z połączonych siecią lokalnie zainstalowanych kontenerów nazwanych *MobileSpaces Runtime Systems*. Cały system agentowy, jak i dostępne serwisy zostały zaimplementowane w języku Java. Wspomniany system udostępnia także wiele specjalnie dla niego stworzonych aplikacji, takich jak edytor dokumentów, przeglądarka obrazków, program do rysowania, zegar, a nawet program pocztowy, i nie przewiduje wykorzystania innych powszechnie dostępnych aplikacji tego typu. Czyni to go systemem zamkniętym, trudnym do rozbudowy i ogranicza jego możliwości powszechnego zastosowania.

Sposobem na uczynienie systemu zarządzającego dokumentami otwartym jest zastosowanie oprogramowania będącego warstwą pośredniczącą (ang. *middleware*) pomiędzy dokumentami-agentami a powszechnymi aplikacjami wykorzystywanymi w pracy nad dokumentem. Cianciarini i in. [CTZ02] prezentują kilka systemów realizujących koncepcję warstwy pośredniczącej dla aktywnych dokumentów. W zasadzie rozszerzają one model Linda [CGMS94] umożliwiający współbieżne przetwarzanie danych poprzez jego implementację w technologii XML (Extensible Markup Language) [BPSM$^+$08]. W Lindzie komunikacja między współbieżnymi procesami

realizowana jest w oparciu o koncepcję współdzielonej przestrzeni danych, zwanej przestrzenią krotek (ang. *tuple space*).

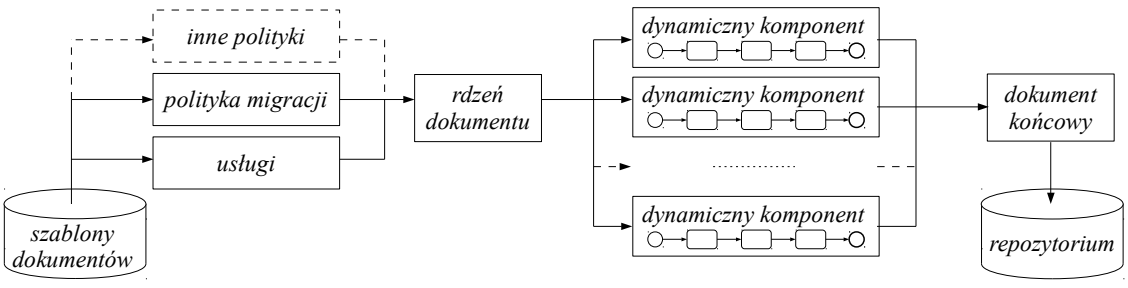
Technologia XML pozwala na tworzenie systemów skoncentrowanych na dokumentach, szczególnie wykorzystujących koncepcję aktywnych i mobilnych dokumentów, gdyż zapewnia łatwe przetwarzanie dokumentów oraz przenośność zawartych w nich treści. Ponadto format XML jest otwarty i można w nim określać nowe typy zawartości oraz definiować złożone struktury danych wymagane przez aplikacje, a także jest ustandaryzowany, co pozwala na wymianę danych między podmiotami mającymi różne pochodzenie.

Cytowani autorzy przeanalizowali różne systemy pośredniczące w koordynowaniu czynności wykonywanych na dokumentach w środowisku rozproszonym, i wybrali z nich cechy „idealnej" warstwy pośredniczącej [CTZ02]. Powinna ona zapewniać następujące możliwości: udostępnianie pewnego zakresu funkcjonalności dołączonej do agentów dokumentu, zarządzanie mobilnością dokumentu, a także koordynowanie zależności między dokumentami w systemie rozproszonym. Warstwa pośrednicząca może być rozproszona i znajdować się na różnych węzłach obliczeniowych lub na jednym dedykowanym serwerze. Jednak oba warianty wymagają implementacji mechanizmów komunikacji dokumentów i aplikacji z warstwą pośredniczącą w celu koordynowania wykonywanych operacji. Czyni to dokumenty ściśle zależnymi od centralnej jednostki zarządzającej nimi oraz od połączenia internetowego.

1.3. Mobilne dokumenty interaktywne (MIND)

Rozprawa proponuje nową koncepcję *Mobilnych i INteraktywnych Dokumentów (MIND)*, wykorzystującą model otwartej architektury obiektowej i umożliwiającą efektywne zarządzanie informacją w dynamicznych wirtualnych organizacjach. Jej szczegóły i sposoby realizacji były już wcześniej opisane w publikacjach [GW09, GW10, God10a, God12].

Cykl życia dokumentu MIND jest przedstawiony na rysunku 1.2. Podstawowym



Rys. 1.2. Cykl życia dokumentu MIND

i niezmiennym komponentem dokumentu, definiującym jego strukturę i zachowanie jest *rdzeń dokumentu* (ang. *hub-document*). Jest on zapisany w języku XML i tworzony na podstawie *szablonów dokumentów*, z których pobierane są informacje o jego strukturze, użytkownikach i czynnościach, które mają być na nim wykonane w środowisku rozproszonym w ramach pracy grupowej, jak również o usługach i politykach określających funkcjonalność dokumentu. Dokument jest podzielony na części (dokumenty składowe), których zawartość może być dowolnego typu (opisanego przykładowo przez typ zawartości MIME). Części przekształcane są następnie na *dynamiczne komponenty*, będące autonomicznymi obiektami zdolnymi do samodzielnej migracji w systemie rozproszonym. Komponenty posiadają wbudowaną *politykę migracji*, którą realizują w sposób równoległy i niezależny, a także *usługi* umożliwiające interakcję z użytkownikiem oraz wykonywanie pewnych czynności automatycznie. Mogą posiadać także *inne polityki*, pozwalające dostosować funkcjonalność dokumentu do potrzeb konkretnej organizacji. Liczba komponentów może się dynamicznie zmieniać poprzez dodawanie lub usuwanie części dokumentu. Architektura jest więc otwarta zarówno dla nowych typów zawartości jak i funkcjonalność, która może być dołączana do dokumentu w formie usług. Zgodnie z przedstawioną wcześniej klasyfikacją, dokumenty MIND można skategoryzować jako wielopostaciowe, łączące zawartość różnego typu oraz funkcjonalność, a także proaktywne (autonomiczne) – pełniące aktywną rolę w realizowanym procesie.

Architektura MIND została opracowana w celu realizacji procesów decyzyjnych w *organizacjach opartych na wiedzy* (ang. *knowledge-based organizations*), w których *pracownicy wiedzy* (ang. *knowledge workers*) wymieniają się informacjami i wiedzą poprzez dokumenty elektroniczne za pośrednictwem sieci komputerowej – najczęściej drogą poczty elektronicznej. Organizacje takie mogą działać w terytorialnym rozproszeniu, jej pracownicy zaś pracować na różnych urządzeniach w różnych miejscach – w pracy, w domu czy w podróży. Celem działania organizacji opartych na wiedzy jest podejmowanie świadomych decyzji (ang. *informed decisions*), będących efektem procesu obliczeń zespołowych, w których umysł ludzki stanowi istotne zewnętrzne źródło wiedzy, ale który bez dostępu do wiedzy zgromadzonej w repozytorium, nie byłby w stanie podjąć takiej decyzji. Problemy decyzyjne rozwiązywane przez organizacje można określić jako niealgorytmiczne, ponieważ decyzje podejmowane są przez ludzi (tj. umysł ludzki jest istotną częścią procesu) w warunkach niepewności, niekompletnej i dynamicznie zmieniającej się informacji oraz niepełnej wiedzy [God09]. Dokumenty MIND wspierają proces decyzyjny poprzez interakcję z pracownikami wiedzy pozwalającą na szybsze wydobywanie wiedzy, automatyzację pewnych formalnych czynności oraz realizację kroków procesu, którego struktura pozostaje zaszyta w polityce migracji wbudowanej

w każdy komponent dokumentu.

Architektura MIND. zaproponowana w rozprawie, umożliwia realizację tezy pracy, która brzmi:

TEZA: Model otwartej architektury obiektowej umożliwia efektywne zarządzanie informacją w dynamicznych wirtualnych organizacjach przy rozwiązywaniu złożonych, niealgorytmicznych problemów decyzyjnych.

W kolejnych rozdziałach rozprawy wykażę tak postawioną tezę, dokonując całościowej analizy funkcjonowania złożonych organizacji i rozwiązywanych przez nie problemów, a także opiszę szczegółowo proponowaną architekturę MIND. Główna idea mojego wywodu polega na wykazaniu, że ta architektura umożliwia realizację kanonicznego zbioru wzorców przepływu pracy w organizacjach, zdefiniowanego dla uogólnionego podejścia procesowo-biznesowego [RHAM06].

Rozdział 2. Rozprawy dokonuje charakterystyki organizacji opartych na wiedzy, dla których została opracowana architektura MIND, gdyż wymiana dokumentów pomiędzy pracownikami zgodnie z ustaloną procedurą stanowi istotę jej działania. MIND umożliwia efektywne zarządzanie informacją w wirtualnych organizacjach, w których pracownicy znajdują się w terytorialnym rozproszeniu i łączą poprzez sieć informatyczną. Dokumenty bowiem zdolne są do samodzielnego sterowania swoim obiegiem zgodnie z ustaloną procedurą przepływu pracy (ang. *workflow*), a także niosą ze sobą informację o czynnościach, które powinny być na nich wykonane w danym punkcie procesu oraz dają możliwość interakcji z użytkownikami w celu ułatwienia wykonania tych czynności. Zapewnia realizację procesu również w sytuacji, gdy pracownicy zmieniają swoją lokalizację i urządzenia, na których pracują lub gdy pojawiają się nowe dokumenty i nowi pracownicy, a także gdy sam proces ulega zmianom w trakcie wykonywania.

Głównym celem organizacji opartych na wiedzy jest rozwiązywanie złożonych, niealgorytmicznych problemów decyzyjnych. Charakterystyka takich problemów jest opisana w rozdziale 3. rozprawy. Ich złożoność wynika ze struktury procesów, które niejednokrotnie składają się z podprocesów (czasem realizowanych poza organizacją), są rozbudowane, obejmujące wiele osób i czynności, a także są oparte na dokumencie jako nośniku wiedzy. Niealgorytmiczność wynika z konieczności interakcji systemu z człowiekiem, który stanowi kluczową rolę w procesie oraz jest źródłem wiedzy zmieniającej się w sposób dynamiczny i nie podlegającym kontroli przez wspierający go system. Zatem głównym celem architektury MIND jest wsparcie człowieka w procesie decyzyjnym, a nie zastąpienie go przez automat.

Rozdział 4. szczegółowo opisuje model architektury MIND i jego poszczególne

moduły. Architektura ta jest obiektowa, gdyż komponenty dokumentu przyjmują postać mobilnych i autonomicznych agentów, stając się dzięki temu dokumentami proaktywnymi, interaktywnymi i zdolnymi do zmieniania swoich lokalizacji. Jest też otwarta, ponieważ jej realizacja nie jest uzależniona od żadnej technologii czy też systemu operacyjnego i pozwala na wykorzystanie istniejących i sprawdzonych narzędzi. Poza tym pozwala na dodawanie dowolnej funkcjonalności w postaci polityk potrzebnych do zarządzania informacją, usług oraz nowych komponentów.

Rozdział 5. proponuje natomiast mechanizm zarządzania przepływem dokumentów MIND oraz wykazuje, że architektura MIND umożliwia efektywną realizację wzorców przepływu dokumentów, występujących w procesach rzeczywistych organizacjach.

Rozdział 6. opisuje sposoby realizacji architektury MIND z wykorzystaniem dostępnych technologii oraz prezentuje wnioski wypływające z walidacji proponowanej architektury. Realizowalność architektury MIND została wykazana poprzez zamodelowanie wzorców przepływu dokumentu w środowisku wykonawczym przepływów pracy oraz za pomocą dwóch prototypów systemu dokumentów mobilnych: jednego wykorzystującego środowisko istniejącej platformy agentowej i drugiego, wykorzystującego opracowany w rozprawie mechanizm klienta pocztowego do zarządzania przepływem dokumentów.

Rozdział 2

Organizacje oparte na wiedzy

Na przełomie lat 60. i 70. pojawiła się nowa tendencja w rozwoju gospodarczym, zmniejszająca znaczenie produkcji fabrycznej na rzecz kompletowania i przetwarzania *informacji* skutkującego gromadzeniem i wykorzystywaniem *wiedzy* [Bel73]. Zaczęły się wówczas kształtować pierwsze organizacje oparte na wiedzy (ang. *knowledge-based organizations*), czyli takie, które koncentrują się na procesach z udziałem wiedzy – jej pozyskiwaniu, tworzeniu, przechowywaniu, transferze, ochronie, udostępnianiu i wykorzystywaniu do podejmowania decyzji. Mimo że wyróżnienie organizacji opartych na wiedzy, jako kluczowych dla gospodarki, to tendencja dosyć nowa, można podać przykłady takich organizacji, które działają i rozwijają się od wieków i są to: sądy, konsylia lekarskie i zespoły dochodzeniowe.

Pojęcie informacji związane jest z zasobem danych (znaków, symboli), które reprezentują zarejestrowane fakty. Uporządkowanie i interpretacja danych w oparciu o pewien pierwotny względem nich wzorzec tworzy *informację*, zaś zbiór wzorców występujących w pewnym kontekście – *wiedzę*. Gromadzenie informacji i związane z tym nabieranie przekonań, nie jest równoważne z gromadzeniem wiedzy, ale może przyczyniać się do jej budowania poprzez odkrywanie nowych wzorców interpretacyjnych [LK01].

Każda organizacja charakteryzuje się pracą grupową *pracowników wiedzy* (ang. *knowledge workers*), która odbywa się zgodnie z ustalonymi procedurami i w oparciu o dokumenty, stanowiące nośnik wiedzy. Praca grupowa występuje w sytuacji, gdy jakieś zadanie nie może być zrealizowane indywidualnie i wymaga współpracy większej liczby osób. Ludzie pełnią kluczową rolę w procesie realizowanym przez organizację, ponieważ na bazie swoich doświadczeń oraz doświadczeń samej organizacji potrafią wydobyć odpowiednie informacje i na ich podstawie podjąć decyzje prowadzące do rozwiązania określonego problemu decyzyjnego. Często także konsekwencją takiego rozwiązania jest utworzenie nowego wzorca, rozszerzającego zakres wiedzy posiadanej przez organizację.

Aby praca grupowa była efektywna, wymaga właściwie zaprojektowanej procedury, która określa zadania poszczególnych osób oraz steruje przepływem pracy (ang. *workflow*) pomiędzy tymi osobami. Procedury organizacyjne wynikają z doświadczenia danej organizacji, są często ściśle sformalizowane i dopracowane tak, aby prowadziły do *prawidłowych* (uzyskanych w wyniku właściwie przeprowadzonej procedury) rozwiązań. W wielu przypadkach procedury podejmowane przez organizacje charakteryzują się także zaufaniem społecznym, czyli przekonaniem, że jeśli postępuje się zgodnie z ustalonymi regułami, to otrzyma się rozwiązanie w danym przypadku najlepsze. Dotyczy to, chociażby, procedur medycznych czy sądowych.

Procesy realizowane przez organizacje oparte na wiedzy w trybie opisanym powyżej nazywać będę *procesami wiedzy* (ang. *knowledge process*). Proces wiedzy to proces biznesowy, którego celem jest wytworzenie nowej wiedzy na podstawie dostępnej informacji i wiedzy zgodnie z ustalonymi regułami jej przepływu. Do wytworzenia nowej wiedzy kluczowa jest rola człowieka – pracownika wiedzy, który pracuje w trybie obliczeń zespołowych, a także struktura procesu i wiedza zgromadzona w repozytoriach (archiwach).

Wraz z rozwojem technologii informacyjnych, powstała możliwość tworzenia wirtualnych organizacji (ang. *virtual organizations*) opartych na wiedzy, których celem jest przeprowadzenie procesu wiedzy w trybie obliczeń zespołowych, przy czym pracownicy wiedzy znajdują się w terytorialnym oddaleniu, kontaktując się ze sobą przy użyciu komputerów lub innych urządzeń elektronicznych o zbliżonych możliwościach, takich jak palmtopy czy smartfony [Mik06]. Wiedza jest wówczas przekazywana za pomocą dokumentów elektronicznych zgodnie z pewną ustaloną polityką ich przepływu, zaś terytorialne oddalenie współpracujących osób nie stanowi bariery w prowadzeniu wspólnych przedsięwzięć.

Najczęstszym kanałem przekazywania dokumentów w wirtualnych organizacjach jest poczta elektroniczna, która umożliwia łatwe przesyłanie dokumentów różnego typu będąc powszechnie akceptowanym i budzącym zaufanie środkiem komunikacji. Nie zapewnia jednak realizacji struktury procesu decyzyjnego, więc decyzja o tym, co zrobić z otrzymaną wiadomością i komu ją przekazać dalej leży w kompetencji pracownika, do którego dana wiadomość została wysłana.

Rozdział ten zawiera charakterystykę organizacji opartych na wiedzy, dla których opracowałam architekturę mobilnych dokumentów interaktywnych MIND. Charakteryzuje organizacje sieciowe, w których komunikacja w głównej mierze opiera się o pocztę elektroniczną. Wyszczególnia zasoby wiedzy organizacji, sposoby ich wzajemnej interakcji oraz istotną rolę ustrukturyzowanej wymiany dokumentów w organizacjach. Na zakończenie proponuje opracowany przeze mnie model wirtualnej organizacji opartej na wiedzy.

2.1. Organizacje sieciowe

Wiadomości tekstowe wysyłane drogą elektroniczną to aktualnie najpopularniejszy sposób komunikacji międzyludzkiej zarówno w miejscu pracy, jak i w życiu prywatnym. Powszechność sieci Internet, jak i upowszechnienie urządzeń mobilnych (laptopów, smartfonów) umożliwiły dostęp do wiadomości niemal w każdym miejscu i w każdym czasie. Badania przeprowadzone w 2011 roku przez firmę telekomunikacyjną Ericsson wykazały, że 35% użytkowników smartfonów odbiera pocztę elektroniczną lub sprawdza wiadomości na Facebooku rano, jeszcze przed wstaniem z łóżka, zaś 40% użytkowników tych telefonów sprawdza wiadomości tuż przed zaśnięciem [Gre11]. Łatwy dostęp do poczty elektronicznej rozmywa także granice czasu, w którym człowiek jest w pracy i poza nią.

Komunikacja za pośrednictwem poczty elektronicznej jest bardzo rozpowszechniona w miejscach pracy już od kilku dekad, zwłaszcza w organizacjach sieciowych, w których pracownicy znajdują się w rozproszeniu geograficznym lub wynikającym z wewnętrznej struktury organizacji. Podstawowymi składowymi takiej organizacji są węzły (ang. *nodes*) i powiązania między nimi (ang. *links*). Węzły reprezentują niezależnych pracowników, którzy komunikują się ze swoimi współpracownikami poprzez pocztę elektroniczną. Każdy pracownik posiada swoją skrzynkę pocztową i jest identyfikowany poprzez adres email. Połączenia nie są ściśle ustalone, powstają dynamicznie poprzez utworzenie pary adresów: odbiorcy i nadawcy, czyli pól `From:` i `To:` w nagłówku wiadomości [FB96].

Popularność poczty elektronicznej wynika z tego, że jest to rozwiązanie:

- tanie, wysyłanie wiadomości często nic nie kosztuje użytkownika,
- szybkie, wiadomość dociera praktycznie natychmiast, niezależnie od odległości,
- wygodne, tworzenie wiadomości jest łatwe, można dołączać załączniki dowolnego formatu niemal jednym kliknięciem, a także dodawać wielu odbiorców,
- asynchroniczne, odbiorca może przeczytać i odpowiedzieć na wiadomość w dogodnym czasie lub zarchiwizować ją do późniejszego wykorzystania,
- ustandaryzowane, powszechność formatu MIME sprawia, że wiadomość będzie czytelna dla każdego odbiorcy, niezależnie od używanego sprzętu i systemu.

Jak zbadali Dabbish i Kraut [DK06], poczta elektroniczna pełni w organizacjach rolę wykraczającą poza zakres jej pierwotnego przeznaczenia. Obok wymiany informacji, a dokładniej przekazywania treści w formie dokumentów, jest wykorzystywana również do przechowywania i archiwizacji dokumentów oraz do koordynowania wykonywanych czynności. W wiadomościach często zawarte są informacje o przyszłych

czynnościach, np. terminach (ang. *deadlines*) lub zdarzeniach, odbiorcy nie usuwają ich natychmiast po przeczytaniu. Autorzy zauważyli również, że wprowadzenie dodatkowych kanałów komunikacyjnych (spotkań, telekonferencji, rozmów telefonicznych) dodatkowo zwiększa intensywność komunikacji poprzez pocztę elektroniczną. Dzieje się tak z powodu potrzeby udokumentowania ustaleń ustnych. Niewątpliwie, poczta elektroniczna jest dominującym narzędziem komunikacyjnym w organizacjach.

Także wielu pracowników uważa pocztę elektroniczną za istotne narzędzie pracy, odczuwając zarazem stres związany z przeciążeniem informacją (ang. *information overload*), z powodu nadmiaru wiadomości, które muszą przetworzyć. Ten nadmiar informacji wpływa też negatywnie na organizację pracy i skuteczność wykonywania poszczególnych czynności przez pracowników. Wykonywane zadania są bowiem często przerywane na skutek napływu kolejnych wiadomości. Psychologowie wskazują, że umysł ludzki z natury nie jest wielozadaniowy i osoba, która co chwilę przełącza uwagę z jednego zadania na drugie ma problemy z koncentracją i pamięcią krótkotrwałą [Gre11]. Badania przeprowadzone w wielu organizacjach wykazują, iż powrót do wykonywanej czynności przez pracownika wiedzy po 30 sekundowym przerwaniu, zajmuje średnio 5 minut [Spi11].

Te powszechne i często niezauważane w organizacjach problemy przeciążenia informacją, w ogólności rodzą poważne problemy ekonomiczne, dające się przedstawić w konkretnych liczbach. Spira [Spi11] przytacza badania przeprowadzone w USA, które wykazały, że koszty przeciążenia informacją wśród 78,6 miliona pracowników wiedzy tego kraju, jak oszacowano w roku 2010, sięgają rocznie 997 miliardów dolarów, co odpowiada ok 28 miliardom roboczogodzin dodatkowego nakładu pracy. Autor obliczył także, że aż 58% pracowników administracji państwowej spędza połowę dnia w pracy na segregowaniu, usuwaniu i sortowaniu wiadomości, co wiąże się z konkretnymi kosztami administracyjnymi. Przeciążenie informacją działa również niekorzystnie na samopoczucie pracowników – te same badania wykazały bowiem, że 66% osób uznało, że nie jest wstanie wykonać wszystkich swoich zadań w standardowym czasie pracy. Co więcej, większości z tych osób zdarzyło się odczuwać psychiczne przytłoczenie na skutek zbyt dużej liczby otrzymywanych wiadomości.

Dlatego pracownicy, próbując radzić sobie z tym problemem, adaptują samodzielnie systemy pocztowe według własnych przyzwyczajeń jako narzędzia zarządzania zadaniami. Są to jednak praktyki niesystemowe i prowizoryczne, a co więcej, dające czasami skutek odwrotny od oczekiwanego, a więc dodatkowo potęgujące odczucie przeciążenia informacją. Przeciętny użytkownik programu pocztowego spędza bowiem ponad 20% czasu na organizowanie lub wyszukiwanie treści wiadomości [BDH+05]. Pracownicy także mają tendencję do arbitralnego nadawania prioryte-

tów wysyłanym wiadomościom (notorycznie zawyżanych), co może przyczyniać się do tego, że odbiorcy czytają wiadomości w niewłaściwej kolejności lub nie czytają ich wcale [WDK11].

2.1.1. Taktyki zarządzania pocztą elektroniczną

Pracownicy stosują różnorodne taktyki posługiwania się pocztą elektroniczną, aby zorganizować swoją pracę, przez co rozszerzają ponad miarę pierwotne przeznaczenie narzędzi pocztowych. Dabbish i Kraut [DK06] zidentyfikowali osiem typowych taktyk stosowanych przez pracowników wiedzy do zarządzania pocztą elektroniczną. Są to:

1. odczytywanie wiadomości, gdy tylko rozbrzmiewa sygnał o jej nadejściu,
2. sprawdzanie wiadomości w określonych, konkretnych odstępach czasu,
3. utrzymywanie jak najmniejszego rozmiaru katalogu `inbox`,
4. przechowywanie wiadomości w katalogu `inbox` jako przypomnienia zadań do zrobienia,
5. pozostawienie wiadomości w katalogu `inbox` po jej przeczytaniu,
6. usuwanie każdej wiadomości związanej z wykonywaną czynnością zaraz po jej przeczytaniu,
7. ręczne katalogowanie wiadomości zaraz po jej odebraniu,
8. archiwizowanie poszczególnych wiadomości w odrębnych katalogach.

Badania nad tymi taktykami doprowadziły cytowanych wcześniej autorów do ciekawych wniosków. Przykładowo, sprawdzanie wiadomości zaraz po usłyszeniu sygnału jej nadejścia, mimo że prowadzi do częstszych przerw w pracy, ma mniejszy wpływ na poczucie przeciążenia informacją, niż sprawdzanie wiadomości o określonych porach. Posiadanie rozbudowanych struktur katalogów również zwiększa poczucie przeciążenia, zaś dążenie do zachowania jak najmniejszego rozmiaru skrzynki odbiorczej wpływa korzystnie na samopoczucie pracownika.

Używając różnych kombinacji tych taktyk, można podzielić pracowników na trzy grupy w zależności od ich osobistych praktyk i przyzwyczajeń [WS96]:

1. nie segregują wiadomości (ang. *no fillers*) – to osoby, które nie segregują wiadomości w osobnych katalogach i trzymają większość wiadomości w katalogu `inbox`; aby zmniejszyć jego rozmiar, sporadycznie usuwają stare wiadomości lub je archiwizują,

2. często segregują wiadomości (ang. *frequent fillers*) – to osoby, które dbają o to, aby ich skrzynka odbiorcza zawierała jak najmniej wiadomości, dlatego bardzo często (każdego dnia) segregują je, przenosząc do odpowiednich katalogów i usuwają nieaktualne,

3. sporadycznie czyszczą wiadomości (ang. *spring cleaners*) – to osoby, które akceptują nadmiarową liczbę wiadomości w skrzynce pocztowej, segregują je do osobnych katalogów od czasu do czasu (średnio co 1-3 miesiące).

Zarządzanie pocztą elektroniczną zależy też od treści wiadomości. Spira [Spi11] argumentuje, że pewne dobre praktyki związane z tworzeniem wiadomości elektronicznych mogą ułatwić organizację pracy i zmniejszyć uczucie stresu związanego z przeciążeniem informacją. Jego wskazówki dotyczące komunikacji poprzez pocztę elektroniczną to:

1. powściągliwość w komunikacji – uważne wybieranie adresatów, unikanie odpowiedzi „do wszystkich",

2. pisanie w sposób jasny – precyzyjnie opisywanie zawartość w tytule, wyróżnianie ważnych informacji, unikanie zbyt długich zdań czy niezrozumiałych skrótów,

3. czytanie tego, co się napisało przed wysłaniem – pozwala to uniknąć zbędnej komunikacji i nieporozumień (na podstawie własnych obserwacji dodam, że przed wysłaniem wiadomości warto sprawdzić, czy zamieściło się odpowiednie załączniki),

4. dokładne czytanie całej otrzymanej wiadomości, zanim wyśle się odpowiedź – zdarza się, że nadawca umieszcza istotne punkty wiadomości na końcu,

5. unikanie umieszczania w pojedynczej wiadomości wielu zagadnień, bo mogą być przeoczone – jedna wiadomość powinna zawierać jeden wątek tematyczny bez pobocznych informacji, jak np. propozycja wspólnego wyjścia na obiad,

6. wiadomości powinny być tak krótkie, jak to tylko możliwe,

7. szanowanie czasu współpracowników, jak swojego własnego – jeśli sprawa nie wymaga natychmiastowej odpowiedzi lub odbiorca jej nie oczekuje, nadawca nie powinien dzwonić ani wysyłać wiadomości przypominających,

8. unikanie niepotrzebnych jednosłownych wiadomości typu „dziękuję" czy „świetnie", nie zawierających praktycznie żadnej informacji.

Wszystkie wymienione taktyki stanowią jednak tylko substytut dla systemowych mechanizmów zarządzania zadaniami pracowników wiedzy. Zapewniają one raczej lokalny komfort pracy poszczególnych osób niż pozwalają na koordynację czynności wykonywanych w skali całej organizacji.

2.1.2. Narzędzia zarządzające pocztą elektroniczną

Personalne taktyki zarządzania pocztą elektroniczną pozwalają pracownikom wiedzy uczynić ją bardziej odpowiednią do charakteru ich pracy. Zarządzanie wiadomościami wspierają narzędzia do obsługi poczty elektronicznej, tj. Microsoft Outlook – popularna komercyjna aplikacja pocztowa, Thunderbird – darmowy klient pocztowy, a także serwisy dostępne przez przeglądarkę internetową, np. Gmail, Yahoo czy Hotmail [TNSV11]. Narzędzia te oferują standardowo możliwość sortowania wiadomości wg nadawcy, odbiorcy albo dacie wysłania, wyszukiwania słów kluczowych w treści wiadomości lub nagłówkach, filtrowania wiadomości według wzorców zdefiniowanych przez użytkownika, a także indeksowania tekstu oraz dodawania znaczników użytkownika. Wsparcie użytkowników dotyczy analizy poszczególnych wiadomości oraz całej skrzynki odbiorczej.

Funkcjonalność lokalnie zainstalowanych klientów pocztowych może być rozszerzona przez dodatkowe aplikacje, np. Xobni [Gre07], NEO[Cas10] czy ClearContext [Mac10] dedykowane dla programu Microsoft Outlook bądź TaQuilla [Mes12] dla Thunderbird. *Xobni* umożliwia klientowi pocztowemu lepszą nawigację wewnątrz skrzynki odbiorczej w oparciu o dodatkowe informacje o nadawcach, automatycznie tworząc ich profile na bazie statystyk, relacji, danych kontaktowych, wątków konwersacji, współdzielonych załączników, a także danych zaczerpniętych z serwisów społecznościowych, takich jak Facebook, Twitter czy LinkedIn. *Nelson Email Organizer* (NEO) rozszerza możliwości programu pocztowego o automatyczną organizację wiadomości w wirtualnych folderach, kategoryzując je według dat, nadawców i załączników. Oferuje także różne widoki w tych folderach oraz możliwość pełnego wyszukiwania słów kluczowych. *TaQuilla* to rozszerzenie pozwalające na automatyczną kategoryzację przychodzących wiadomości z wykorzystaniem bayesowskiej analizy zbioru znaczników wiadomości już odebranych – technice wprowadzonej wcześniej do wykrywania i eliminacji spamu.

Bardziej zaawansowanym rozwiązaniem jest *ClearContext* wspierający koordynowanie przepływu pracy z wykorzystaniem klienta pocztowego Microsoft Outlook. Umożliwia on zarządzanie zdarzeniami poprzez tworzenie przepływu pracy na podstawie informacji dostępnych w wiadomościach pocztowych. Potrafi wydobywać daty z wiadomości, priorytetyzować nadawców, przekazywać zadania i wiadomości

do współpracowników lub odraczać dostarczenie wiadomości, dopóki użytkownik nie będzie w stanie jej przetworzyć. Przepływ pracy jest w tym przypadku ustalany na bieżąco i lokalnie, a zatem nie realizuje globalnie zaprojektowanej procedury organizacyjnej.

Together Workflow Server (TWS) [Tog11a] daje możliwość integracji przepływu pracy z programem Microsoft Outlook. Przepływ pracy jest wówczas dostarczany jako zewnętrzna usługa poprzez połączenie z aplikacją TWS i pozwala zarządzać listą zadań za pośrednictwem Outlooka. W tym przypadku koordynowanie zadań może odbywać się poprzez zaprojektowany globalnie przepływ pracy, ale nie wynika on wówczas z treści samych wiadomości. Zatem przesyłane dokumenty pełnią jedynie statyczną rolę nośnika informacji w procesie.

Śledzenie wielu równocześnie realizowanych zadań przez jednego pracownika pracującego z dokumentami przekazywanymi przez pocztę elektroniczną jest trudne, męczące i podatne na błędy, gdyż dzisiejsze programy pocztowe nie zapewniają wystarczającej funkcjonalności do zarządzania zadaniami. Bellotti i in. w [BDH+05] sformułowali, zbiór wymagań dla programu pocztowego, wspierającego pracę grupową w oparciu o dostarczane wiadomości. Program taki powinien:

- dawać możliwość uruchamiania określonych aplikacji, kiedy są potrzebne,
- trzymać wiadomości dotyczące jednego wątku w jednym miejscu wraz z ich historią,
- grupować wiadomości dotyczące jednej czynności,
- łączyć różne informacje i zasoby z czynnościami, a nie umiejscawiać ich w osobnych katalogach aplikacji (np. dane kontaktowe czy załączniki zazwyczaj są umieszczane w odrębnych katalogach nie związanym z czynnościami),
- umożliwiać zarządzanie wiadomościami poprzez: tworzenie list rzeczy do zrobienia, wprowadzania ograniczeń czasowych i przypomnień.

Z punktu widzenia tezy tej rozprawy stawiam także postulat, że program pocztowy powinien zapewniać wsparcie dla realizacji przepływu pracy wynikającego z procedury organizacyjnej w oparciu o dokumenty. Aktualnie bowiem, pomimo różnych funkcjonalności dostarczanych przez klientów pocztowych, wiele czynności związanych z zarządzaniem dokumentami musi być wykonywana manualnie przez pracowników wiedzy. W szczególności, programy pocztowe nie zapewniają wsparcia dla różnych *wzorców współpracy*, które są silnie zależne od semantyki dokumentów.

2.2. Zasoby wiedzy organizacji

Komunikacja poprzez pocztę elektroniczną w organizacjach sieciowych i taktyki zarządzania wiadomościami przez pracowników wskazują, że w celu realizacji zadań organizacji niezbędne są trzy elementy:

- treść wiadomości zawierająca informacje i zadania,
- możliwość zarządzania czynnościami,
- obecność pracowników, którzy przetwarzają otrzymane informacje i wykonują określone zadania.

Istotą organizacji opartych na wiedzy jest to, że pracownicy, pracując w zespołach, wytwarzają nową wiedzę dzięki nieustannej wymianie informacji między sobą za pośrednictwem dokumentów. Celem działania organizacji opartych na wiedzy jest zatem realizacja procesów w trybie obliczeń zespołowych, w których umysł ludzki stanowi istotny zasób. W ogólności w organizacjach wyróżniamy trzy rodzaje zasobów wiedzy: wiedzę *ugruntowaną*, *skodyfikowaną* i *spersonalizowaną* [Mik06].

1. **Wiedza ugruntowana** to wypracowane procedury organizacyjne. Nadają one strukturę procesowi wiedzy i stanowią wiedzę *niejawną* organizacji. Procedury tworzone są na podstawie zewnętrznych przepisów ustalanych, przykładowo, przez organy ustawodawcze lub wynikają z wieloletnich doświadczeń organizacji, stanowiąc kwintesencję dobrych praktyk w niej stosowanych. Mają na celu zagwarantowanie przeprowadzenia wszystkich czynności niezbędnych do uzyskania *prawidłowego rozwiązania* (warunki prawidłowego rozwiązania przedstawię w rozdziale 3).

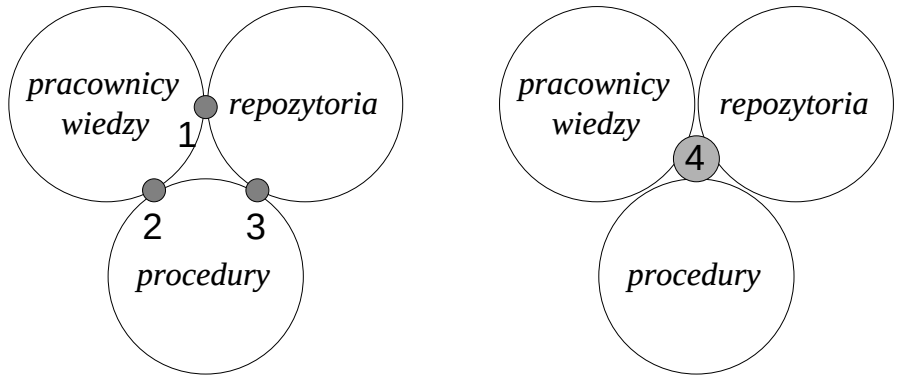
2. **Wiedza skodyfikowana** to repozytoria dokumentów przechowujące *jawną* wiedzę organizacji. Są to dokumentacje procesów oraz definicje typów dokumentów przechowywane w formie umożliwiającej efektywne wyszukiwanie, kategoryzację i wydobywanie danych, np. w bazach danych, bibliotekach cyfrowych czy bazach wiedzy.

3. **Wiedza spersonalizowana** to wiedza pracowników organizacji, którą można określić jako wiedzę *ukrytą* organizacji, gdyż zazwyczaj nie jest nigdzie zarejestrowana. Rolą pracowników jest wytworzenie nowej wiedzy w oparciu o wiedzę skodyfikowaną i zgodnie z ustalonymi procedurami organizacji. Pracownicy tworzą w trakcie procesu kolejne jednostki wiedzy i przekazują je następnym pracownikom za pośrednictwem dokumentów. Tu właśnie uwidacznia

się znacząca rola człowieka w podejmowaniu decyzji i rozwiązywaniu problemów, dla których nie istnieje (albo przynajmniej nie jest znane) rozwiązanie algorytmiczne.

Wymienione trzy rodzaje zasobów wiedzy organizacji są ze sobą ściśle powiązane i oddziałują na siebie, przy czym wiedza spersonalizowana (pracowników wiedzy) jest zasobem kluczowym. Mechanizmy interakcji powinny umożliwić pracownikom dostęp do trzech wymienionych zasobów, a więc powinny zapewniać efektywną *komunikację* treści, czyli wymianę jednostek informacji, jak i *koordynację* czynności, które ma wykonać człowiek w ramach procesu wiedzy.

Do implementacji interfejsów pośredniczących między zasobami wiedzy organizacji można wykorzystać jeden z dwóch paradygmatów modelowania i implementowania systemów wspomnianych w poprzednim rozdziale: *skoncentrowany na danych* (ang. *data-centric*) lub *skoncentrowany na dokumencie* (ang. *document-centric*). Pierwszy model zakłada interakcję miedzy zasobami z wykorzystaniem identycznych struktur danych do wymiany informacji, drugi model zaś zakłada, że to dokumenty są jednocześnie jednostkami *informacji* i *interakcji*. Różnicę między tymi podejściami ilustrują rysunki 2.1(a) i 2.1(b).



(a) Model skoncentrowany na danych

(b) Model skoncentrowany na dokumentach

Rys. 2.1. Zasoby wiedzy organizacji oraz modele interakcji między nimi

Interfejs pomiędzy pracownikami wiedzy a repozytoriami (punkt „1") zapewnia możliwość formułowania zapytań w celu wyszukiwania odpowiednich informacji w repozytoriach dokumentów. Procedury organizacyjne zawierają informacje o czynnościach, które powinni wykonać uczestnicy pracy grupowej oraz o kolejności tych czynności (punkt „2"), a także o dokumentach, które biorą udział w procesie (punkt „3"), wskazując pracownikom wiedzy, jakie dokumenty powinni odszukać w repozytorium. Jeśli procedury są wykonywane automatycznie, np. jako zaimplementowane

przepływy pracy, wówczas mogą przy pomocy odpowiednich serwisów wyszukiwać potrzebne dane w repozytoriach oraz mogą pośredniczyć między pracownikami, wyznaczając kolejność wykonywanych czynności. Podejście skoncentrowane na danych umożliwia wymianę dokumentów zgodnie z procedurą, ale jego celem jest po prostu przekazywanie informacji.

Inaczej jest w pracy grupowej skoncentrowanej na dokumencie, gdzie dokument może automatycznie uruchamiać odpowiednie aplikacje wymagane do pracy z jego treścią bądź przynajmniej podpowiedzieć użytkownikowi, żeby to zrobił. Jeśli dokumenty są zrozumiałe zarówno dla człowieka, jak i komputera, to mogą stanowić ujednolicony interfejs dla trzech zasobów wiedzy organizacji (punkt „4") na rysunku 2.1(b) oraz pełnić aktywną rolę w procesie wiedzy.

2.3. Rola dokumentów w procesie wiedzy

W organizacjach sieciowych, w których komunikacja w głównej mierze opiera się o pocztę elektroniczną, nie ma bezpośredniego wsparcia dla koordynacji i monitorowania czynności procesowych oraz jawnych powiązań między współpracującymi osobami. Procedury organizacyjne często nie stanowią wyodrębnionego zasobu wiedzy, a raczej wchodzą w skład ukrytej (spersonalizowanej) wiedzy pracowników. To pracownicy, na bazie swoich osobistych taktyk i umiejętności zarządzają czynnościami – tworzą połączenia między innymi pracownikami, wybierając adresatów wiadomości dynamicznie, w chwili tworzenia wiadomości. Taka sytuacja może prowadzić do różnego rodzaju pomyłek: wiadomości przekazywane są do niewłaściwych osób, zbyt późno lub wcale. Dodatkowo kontrola przebiegu procedury jest ograniczona, gdyż nie ma możliwości sprawdzenia, czy odbiorca przeczytał wiadomość i podjął związane z nim działanie, a także trudno wyodrębnić i nadzorować działania wykonywane równolegle przez inne osoby. Do tego dochodzą wymienione szczegółowo w podrozdziale 2.1 negatywne skutki korzystania z poczty elektronicznej, takie jak przeciążenie informacją czy pomijanie niektórych wiadomości.

W organizacjach opartych na wiedzy przestrzeganie procedur jest bardzo istotne, stąd konieczne jest zapewnienie systemowego koordynowania czynności wykonywanych przez pracowników uczestniczących w procesie wiedzy. Koordynacja czynności może być narzucona i egzekwowana przez prawo, zwłaszcza, gdy procedury są wykonywane manualnie przez pracowników wiedzy – taka sytuacja ma miejsce w procesach sądowych czy w przypadku procedur medycznych. Pracownicy są wówczas odpowiedzialni przed prawem za przestrzeganie obowiązujących procedur.

Aktywne dokumenty mogą, jednakże, odciążyć pracowników wiedzy od konieczności samodzielnego koordynowania czynności i pozwolić im się skupić na istocie

swojej pracy, czyli na wytwarzaniu nowej wiedzy i podejmowaniu decyzji. Dokumenty powinny automatyzować część procesu wiedzy poprzez implementację procedur organizacyjnych, a więc migrację dokumentów pomiędzy pracownikami wiedzy oraz wspierać pracowników poprzez interakcję oraz dostosowanie zawartości do ich indywidualnych potrzeb czy aktualnych wymagań urządzeń osobistych Pracownik może wykonywać daną pracę związaną z dokumentem na różnych urządzeniach, np. w pracy, w domu czy w podróży. Zmienia się wówczas kontekst interakcji dokumentu z tym pracownikiem związany z jego aktualnym środowiskiem pracy.

Polityka migracji dokumentu jest opisana przez przepływ pracy. *Przepływ pracy* (ang. *workflow*) to proces, rozumiany jako następstwo rozróżnialnych zdarzeń, podczas których dokumenty, informacje lub zadania są przekazywane od jednego uczestnika do następnego, według odpowiednich procedur zarządczych.

Przepływ pracy definiuje strukturę procesu, która składa się ze zbioru zdarzeń zwanych *czynnościami* (ang. *activities*) oraz skierowanych przejść między nimi, zwanych *tranzycjami* (ang. *transitions*). Czynności są rozróżnialne, czyli można im przypisać unikatowy identyfikator oraz występują w ustalonej kolejności, którą wyznaczają tranzycje. Proces może składać się z *podprocesów*, stanowiących spójny podzbiór czynności i tranzycji, definiowany w ramach czynności procesu nadrzędnego.

Proces realizuje cele i politykę organizacji w ramach zdefiniowanej struktury organizacyjnej. Definiuje dodatkowo pewne warunki wejściowe, które go inicjują oraz wyjściowe, które go kończą, a także możliwe relacje między uczestnikami procesu. Czynności mogą być wykonywane automatycznie lub poprzez pracowników, których wiedza stanowi zewnętrzne źródło danych w procesie [WfM99].

W rozprawie proponuję, by w organizacjach opartych na wiedzy stosować dokumenty, które są zarówno nośnikami wiedzy (treść i struktura dokumentu) i jednostkami interfejsu (funkcjonalność dokumentu), ułatwiającymi wydobywanie wiedzy i automatyczną realizację obiegu dokumentów zgodnie z polityką migracji określoną przez procedury organizacyjne. Proponowany typ dokumentu powinien zatem spełniać trzy podstawowe wymogi, będąc równocześnie:

- interaktywnym, tj. czytelnym, łatwym w edycji, wspierającym proces „wypełniania się" treścią, dostosowującym się do charakteru pracy danej osoby, ułatwiającym lokalną pracę na tym dokumencie,
- mobilnym, tj. realizującym przepływ pracy zgodnie z wbudowaną polityką migracji, przenoszącym informacje kolejnej osobie procesu wiedzy w obrębie pewnej sieci komputerowej,
- kompatybilnym, tj. pozwalającym na odczyt zawartości na dowolnym urządza-

niu oraz zarządzającym prawami dostępu i innymi zabezpieczeniami treści.

Poniżej zdefiniuję na użytek tej rozprawy model organizacji opartej na wiedzy oraz przeanalizuję możliwości wykorzystania w niej rozproszonego i mobilnego dokumentu zaproponowanego wcześniej (zob. rys. 1.2, str. 19).

2.4. Model organizacji opartej na wiedzy

Struktura danej organizacji opartej na wiedzy definiuje specyficzny dla niej *proces wiedzy*, wyznaczający przepływ dokumentów pomiędzy poszczególnymi uczestnikami, czynności wykonywane przez nich na dokumentach oraz typy dokumentów uczestniczących w procesie. Formalnie, model organizacji opartej na wiedzy (OOW) wymaga zdefiniowania trzech zbiorów: *czynności* (A) i *tranzycji* (T), które występują w danym przepływie pracy oraz wymaganych *typów dokumentów* (S), które mogą znajdować się w repozytoriach: ogólnodostępnych (np. typy MIME [FB96]) lub wewnętrznych należących do organizacji i przechowujących szablony dokumentów zdefiniowane dla jej procesów.

Organizacja Oparta na Wiedzy (OOW) składa się zatem ze zbioru czynności A, zbioru tranzycji T ($A \cap T = \emptyset$) oraz zbioru typów dokumentów S:

$$OOW = (A, T, S). \tag{2.1}$$

Na podstawie danych dostępnych w OOW definiowany jest dokument D, składający się z wielu komponentów (dokumentów składowych). Każdy dokument składowy ma przypisany pewien typ ze zbioru typów dokumentów:

$$S = \{s_1, s_2, ..., s_n\}. \tag{2.2}$$

Każdy dokumentu składowy jest więc *egzemplarzem* pewnego typu $s_i \in S$ i w skład dokumentu D może wchodzić wiele egzemplarzy jednego typu.

E jest rodziną zbiorów egzemplarzy typów dokumentu:

$$E = \{e_1, e_2, ..., e_n\}, \tag{2.3}$$

zaś elementami rodziny zbiorów E są zbiory w postaci:

$$e_i = \{e_i^1, e_i^2, ..., e_i^k\}, \tag{2.4}$$

którego elementami są wszystkie egzemplarze jednego typu s_i występujące w dokumencie D.

Funkcja:

$$\sigma : S \rightarrow E \tag{2.5}$$

pozwala przyporządkować danemu typowi dokumentu zbiór jego egzemplarzy, czyli $\sigma(s_i) = e_i$.

Pojedynczym dokumentem składowym dokumentu D jest element e_i^j, którego indeks dolny i określa jego typ, zaś indeks górny j jest identyfikatorem elementu w zbiorze e_i. Indeksy i oraz j tworzą unikatowy identyfikator dokumentu składowego. Zatem dokument D jest zbiorem składającym się z dokumentów składowych e_i^j. Przykładowo w dokumencie $D_1 = \{e_2^4, e_1^6, e_3^1\}$:

- element e_2^4 jest dokumentem składowym, należącym do zbioru e_2 (zbiór egzemplarzy jednego typu, który w rodzinie zbiorów E ma indeks 2) i w tym zbiorze ma indeks 4;
- element e_1^6 jest dokumentem składowym, należącym do zbioru e_1 i mającym w nim indeks 6;
- element e_3^1 jest dokumentem składowym, należącym do zbioru e_3 i mającym w nim indeks 1;

Następujące funkcje określają porządek elementów w zbiorze D:

$$pred : D \rightarrow D \cup \{nil\}, \tag{2.6}$$

$$succ : D \rightarrow D \cup \{nil\}. \tag{2.7}$$

Dokumenty składowe migrują pomiędzy pracownikami wiedzy, realizując w ten sposób proces wiedzy. Zbiór D jest zatem, w trakcie wykonywania procesu, zbiorem fizycznie rozproszonych elementów, które zmieniają swoją lokalizację w czasie. Także moc zbioru D może się zmieniać podczas wykonywania procesu, tzn. składowe mogą być dodawane, usuwane, łączone lub dzielone.

Proces jest opisany grafem dwudzielnym skierowanym, którego węzłami są *czynności* ze zbioru (A) i *tranzycje* ze zbioru (T). Taka reprezentacja wykorzystuje model sieci Petriego[1] [Sta87, JK09].

Czynności mają przypisanych wykonawców – pracowników wiedzy i mogą być realizowane na różnych urządzeniach osobistych. Przy czym pracownicy nie są ściśle przypisani do jednej lokalizacji – mogą pracować na różnych urządzeniach lub korzystać z urządzeń mobilnych w różnych miejscach. Aktualną lokalizację urządzenia, na którym wykonywana jest dana czynność, wyznacza funkcja:

$$\varphi : A \rightarrow IP \tag{2.8}$$

przyporządkowująca pracownikom ze zbioru A adres ze zbioru IP. Zmiana lokalizacji może zmieniać także kontekst wykonywania czynności, związany z zasobami

[1] W definicji sieci Petriego czynności nazywane są miejscami (ang. *places*) [Sta87].

lokalnego systemu danego urządzenia lub sposobem łączenia z Internetem. Element zbioru czynności określa pojedyncze zadanie lub podproces, który jest zawsze zamknięty w obrębie pojedynczej czynności. Zatem model *OOW* może składać się z wielu zagnieżdżonych podprocesów.

Węzły tranzycji rozdzielają węzły czynności i mogą określać dodatkowe warunki przejść między czynnościami. Skierowane krawędzi grafu wyznaczają ścieżkę przepływu i prowadzą od czynności do tranzycji lub od tranzycji do czynności. Przykładowo, jeśli tranzycja t_1 znajdująca się między czynnościami a_1 i a_2, to krawędzi w grafie opisującym proces wyznaczone są przez dwie pary: $(a_1, t_1), (t_1, a_2)$. Krawędzie w procesie tworzą zatem zbiór $L \subseteq A \times T \cup T \times A$.

Odpowiednikami znaczników sieci Petriego w przedstawionym modelu są dokumenty składowe. Różnią się one od znaczników w sieci Petriego tym, że niosą ze sobą informacje (podobnie jak w kolorowanych sieciach Petriego [Jen87]), ale też stanowią jednostkę interfejsu i pełnią aktywną rolę w procesie wiedzy.

Dokument D jest zbiorem dokumentów składowych, które podczas realizacji procesu znajdują się w fizycznym rozproszeniu. Funkcja:

$$\gamma : e_i \rightarrow A \tag{2.9}$$

zwraca aktualną czynność w procesie, w której znajduje się dany jako argument dokument składowy. Wartość funkcji γ jest zatem *stanem lokalnym* dokumentu składowego w procesie. Dla wszystkich elementów zbioru D, wykonanie operacji rzutowania $map(\gamma, D)$ powoduje uzyskanie zbioru wszystkich czynności procesu nadrzędnego (czyli bez uwzględnienia podprocesów wykonywanych w obrębie czynności), w których aktualnie przebywają dokumenty składowe. Zbiór ten opisuje *stan globalny* procesu wiedzy. Jeśli aktualna czynność dokumentu jest podprocesem, to mówimy o *stanie zagnieżdżonym* procesu, który można określić poprzez wyznaczenie stanu globalnego dla tego podprocesu.

Na podstawie zbioru D i zdefiniowanych funkcji można otrzymać także inne informacje, np.:

- wszystkie adresy IP, na których aktualnie znajdują się dokumenty składowe,
- wszystkie aktywne czynności w danej chwili procesu,
- wszystkie dokumenty składowe, które znajdują się u konkretnego autora.

Powyższe informacje są niezbędne do śledzenia przebiegu procesu wiedzy i rozwoju dokumentu rozproszonego D. Jednakże kontrola globalnego stanu procesu i aktualnych lokalizacji w systemie rozproszonym zależna jest od sygnalizowania swojej lokalizacji przez dokumenty składowe. W luźno powiązanych (ang. *loosely coupled*) systemach rozproszonych wyznaczenie rzeczywistego stanu globalnego dla danej chwili

jest trudne do zrealizowania. Z drugiej strony jednak znajomość globalnego stanu takiego systemu nie jest w zasadzie potrzebna, poza wyjątkowymi sytuacjami wymagającymi akcji naprawczych w przypadku zaginięcia przesyłanych dokumentów lub opóźnień w wykonywaniu poszczególnych czynności procesu. Kontrola stanu procesu odbywa się w takich sytuacjach poprzez śledzenie realizacji procesu, czyli sekwencję poszczególnych stanów lokalnych dokumentów składowych. Do tego zagadnienia wrócę w rozdziale 5 rozprawy.

Rozdział 3

Niealgorytmiczne procesy decyzyjne

Pojawienie się komputerów i ich szybki rozwój przyczyniły się do podjęcia prób automatyzacji procesów decyzyjnych. Konieczna była więc ich algorytmizacja. Okazało się jednak, że skonstruowanie algorytmu dla niektórych problemów decyzyjnych jest sprawą trudną (tzn. algorytm jest bądź zbyt złożony, by można go było zaimplementować, bądź nie wiadomo, czy w ogóle istnieje). Postępująca komputeryzacja wprowadziła kanon interakcji człowiek-komputer, co umożliwiło wykorzystanie umysłu ludzkiego w procesie rozwiązywania „trudnych" problemów decyzyjnych. Umysł ludzki, jak na razie, wymyka się algorytmizacji, szczególnie w odniesieniu do intuicji (często wykorzystywanej w pracy dochodzeniowej) czy podejmowania ryzyka (np. biznesowego).

Problemy decyzyjne można zatem podzielić na algorytmiczne i niealgorytmiczne. *Algorytmiczne* problemy decyzyjne to takie, dla których można dowieść, że algorytm istnieje, a więc da się dla nich skonstruować maszynę Turinga [Tur36]. Oczywiście, algorytmy te mogą być w wielu przypadkach złożone, a zatem trudne lub nieopłacalne w realizacji, co jednak nie kwestionuje ich istnienia. Warunkiem zaś możliwości znalezienia algorytmu jest całkowita kontrola nad wszystkimi źródłami danych.

Warunek ten nie jest spełniony w przypadku interakcji systemu z zewnętrznymi źródłami danych, w tym w szczególności w interakcji z człowiekiem. Zatem problemy decyzyjne, w których decyzje podejmowane są przez ludzi (tj. umysł ludzki jest istotną częścią procesu) często w warunkach niepewności, niekompletnej i dynamicznie zmieniającej się informacji oraz niepełnej wiedzy, są problemami *niealgorytmicznymi*. Dla takich problemów nie ma możliwości skonstruowania maszyny Turinga, gdyż model ten jest zamknięty na nowe dane, dynamicznie dodawane z zewnątrz w trakcie realizacji procesu oraz nie umożliwia modelowania

upływu czasu [Weg97]. Można natomiast formalnie opisać procedury rozwiązywania problemów decyzyjnych dla konkretnych organizacji.

Przestrzeganie procedur pełni istotną rolę w procesach decyzyjnych podejmowanych w organizacjach opartych na wiedzy. W procesach takich, zwanych *procesami wiedzy*, istnieje konieczność współpracy wielu specjalistów w warunkach niepełnej wiedzy i niekompletnej informacji, w tym również w sytuacjach konfliktowych. Ich intelekt i doświadczenie, a zatem umiejętność wnioskowania, skuteczna weryfikacja informacji oraz intuicja, w powiązaniu z jawnie narzuconą strukturą organizacyjną (np. hierarchią służbową) oraz dokumentacją procesu, skutkują podjęciem decyzji określanej mianem *prawidłowej*. Rozróżnienie decyzji *prawidłowej* od *poprawnej* jest w tym przypadku bardzo istotne ze względu na działanie w warunkach wiedzy niepełnej i gromadzonej dynamicznie jeszcze w trakcie procesu podejmowania decyzji [God09].

Systemy wpierające rozwiązywanie niealgorytmicznych problemów decyzyjnych tworzą klasę systemów otwartych ze względu na zewnętrzne i dynamicznie zmieniające się źródła danych. Ich kluczowym zadaniem jest przede wszystkim koordynowanie czynności wykonywanych w procesie wiedzy. W procesach tych można wyszczególnić kanoniczny zbiór wzorców przepływu sterowania opracowany na podstawie obserwacji rzeczywistych organizacji [ARH11, RHAM06]. Zbiór ten dostarcza elementy pozwalające na skonstruowanie dowolnie złożonego procesu wiedzy.

W dalszej części rozprawy przedstawiam charakterystykę interaktywnych niealgorytmicznych procesów obliczeniowych, wykazując ich przydatność w poszukiwaniu rozwiązywania problemów decyzyjnych przez zespoły ludzkie. Analizuję również wzorce przepływu pracy, będące elementami składowymi procesów wiedzy z perspektywy organizacji wykorzystujących pocztę elektroniczną jako podstawową formę komunikacji uczestników procesu.

3.1. Niealgorytmiczność interaktywnych procesów obliczeniowych

Współczesne komputery to nie tylko maszyny realizujące algorytmy, ale przede wszystkim jednostki pozwalające na interakcję z użytkownikiem. Interakcja jest umożliwiona dzięki graficznym interfejsom, obiektowym i rozproszonym programom, a także połączeniu użytkowników w różnego rodzaju sieci: korporacyjne, publiczne, mobilne, ad-hoc, itp. W ten sposób użytkownicy oddziałują na przebieg procesu i siebie nawzajem, co umożliwia realizację zadań, które mogą nie mieć rozwiązania algorytmicznego.

Możliwość znalezienia algorytmu rozwiązującego pewien problem jest równoważna z możliwością skonstruowania dla tego problemu maszyny Turinga [Tur36]. Maszyna Turinga przekształca łańcuchy symboli wejściowych zapisane na taśmie w łańcuchy symboli wyjściowych, które odzwierciedlają sekwencję zmiany stanów maszyny. W każdym kroku działania, maszyna odczytuje symbol z określonego pola na taśmie i zależnie od tego symbolu i stanu maszyny zapisuje nową wartość w danym polu, zmienia stan, a następnie może przesunąć się o jedno pole w prawo lub w lewo. Alfabet dopuszczalnych symboli może być dowolnym zbiorem skończonym i nie może się zmieniać po rozpoczęciu obliczeń. Model maszyny Turinga nie zezwala na dodawanie nowych symboli do alfabetu dopuszczalnych symboli, gdy maszyna jest już uruchomiona. Zamyka go to na interakcję, która dostarcza dane z zewnątrz do procesu obliczeniowego. Szczególnie interakcja komputera z człowiekiem (ang. *human–computer interaction*) dostarcza nowe symbole potrzebne do realizacji procesu.

Już pod koniec życia, Turing zaproponował rozszerzenie swojego modelu maszyny o operacje wejściowe i wyjściowe [Tur50], tworząc koncepcję *maszyny interaktywnej*, która wspiera dynamiczną interakcję ze środowiskiem zewnętrznym. Prace te długo pozostawały w zapomnieniu, dopiero za sprawą rozpowszechnienia się systemów interaktywnych, wzbudziły zainteresowanie badaczy [Weg97]. Wcześniej bowiem istniało powszechne przekonanie, że wszystkie problemy obliczalne można rozwiązać algorytmicznie, a więc przy pomocy tradycyjnej maszyny Turinga.

Pojawiło się zatem pytanie, czy działanie maszyny interaktywnej da się zredukować do działania tradycyjnej maszyny Turinga. Zdecydowanie przecząco odpowiada na nie Wegner [Weg97], który przedstawia różnice między procesem realizowanym przez algorytm a procesem opartym na interakcji na przykładzie kontraktu sprzedaży i kontraktu małżeńskiego. Proces algorytmiczny jest jak „kontrakt sprzedaży", w którym rozwiązanie jest zależne od danych wejściowych (element systemu zobowiązuje się *wykonać zadanie* dla danych wejściowych spełniających ustalone wcześniej wymogi). Natomiast proces oparty na interakcji z użytkownikami jest jak „kontrakt małżeński" (element systemu zobowiązuje się do *określonego zachowania* bez sprecyzowania danych wejściowych). Podobnie, jak w praktyce życia społecznego, kontraktu małżeńskiego nie da się sprowadzić do kontraktu sprzedaży.

Zatem wprowadzenie interakcji, a więc niewielkiej zmiany w modelu maszyny Turinga, znacznie zwiększa jej możliwości i czyni zbyt skomplikowaną do wyrażenia jej „przyjaznym" modelem matematycznym. W maszynie interaktywnej pojawiają się bowiem dane wejściowe, nad którymi program nie ma kontroli i mogą one zawierać wcześniej nieznane symbole. Generalnie, różne formy interakcji pozwalają przekształcić system z zamkniętego w otwarty i dzięki temu rozwiązywać problemy wykraczające poza możliwości tradycyjnej maszyny Turinga.

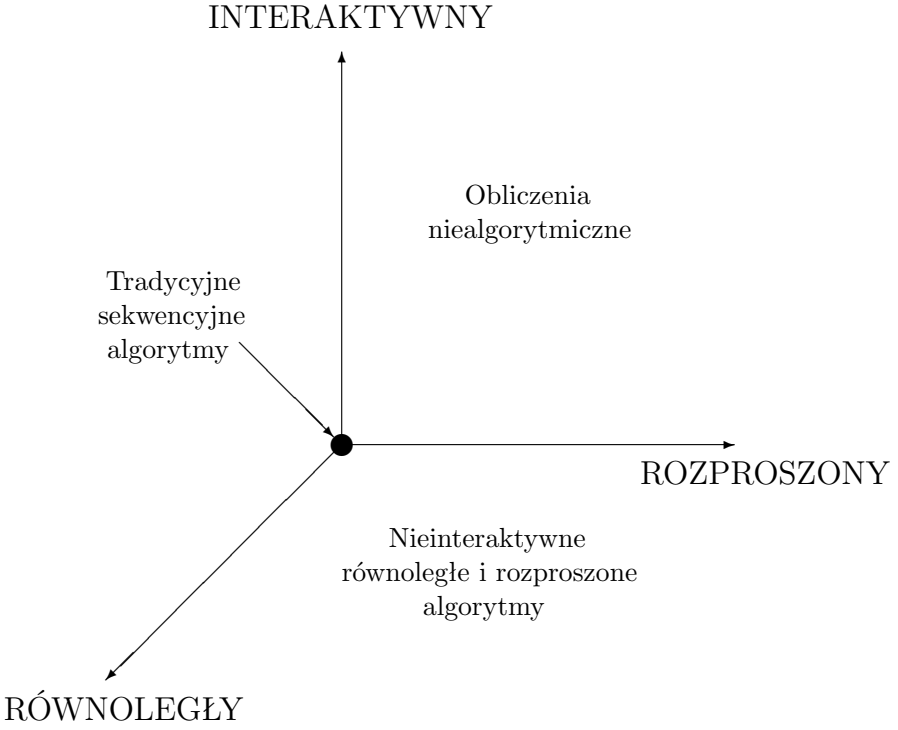
Wegner i Goldin [GSW04, GW05, GW08] uzasadnili, że wprowadzenie interakcji do procesu obliczeniowego znacznie osłabia hipotezę Turinga, zakładającą algorytmiczność wszystkich obliczeń wykonywanych przez komputer. Zaproponowali oni nowy model – *Persistent Turing Machine (PTM)*, będący rozszerzeniem maszyny Turinga o interakcję, powtarzające się odbieranie danych z zewnątrz do czasu otrzymania rozwiązania oraz zapamiętywanie wcześniejszych stanów, czyli kontekstów przetwarzania taśmy. Udowodnili [GSW04], że zaproponowane rozszerzenie wykracza poza możliwości tradycyjnej maszyny Turinga. Pozwoliło im to na sformułowanie nowych tez odpowiadających możliwościom współczesnych komputerów. Wnioski wypływające z hipotezy Turinga oraz nowe sformułowania, biorące pod uwagę możliwości obliczeniowe maszyny interaktywnej prezentuje poniższa tabela:

Tabela 3.1: Możliwości obliczeniowe współczesnych systemów

Wnioski z hipotezy Turinga	**Wnioski zaproponowane przez Wegnera i Goldin**
Wszystkie problemy *obliczalne* można rozwiązać za pomocą funkcji.	Wszystkie problemy *algorytmiczne* można rozwiązać za pomocą funkcji.
Wszystkie problemy *obliczalne* można opisać algorytmem.	Wszystkie problemy *oparte na funkcjach* można opisać algorytmem.
Algorytmy są tym, co mogą wykonywać *dowolne komputery*.	Algorytmy są tym, co mogły wykonywać *wczesne* komputery.
Maszyna Turinga jest ogólnym modelem *dowolnych* komputerów.	Maszyna Turinga jest ogólnym modelem *wczesnych* komputerów.
Maszyna Turinga potrafi zasymulować *dowolny* komputer.	Maszyna Turinga potrafi zasymulować *dowolne urządzenie* realizujące algorytmiczne obliczenia.
:(	Maszyna Turinga nie potrafi rozwiązać każdego problemu obliczeniowego ani wykonać wszystkich operacji, które realizują *współczesne systemy*.

Rysunek 3.1 [Weg97] przedstawia przestrzeń możliwych do zaimplementowania modeli obliczeń. Każdy punkt w tej przestrzeni reprezentuje pewną możliwą do zaimplementowania klasę obliczeń, których różne części (procesy) wykonywane są równolegle, w rozproszeniu bądź w odpowiedzi na zdarzenia zewnętrzne. Klasyczne algorytmy sekwencyjne odpowiadają na rysunku 3.1 środkowi układu. W celu

zwiększenia wydajności obliczeń, algorytmy są często wykonywane równolegle i/lub w rozproszeniu, jednak obliczenia te są nadal algorytmiczne. Model obliczeń jest równoległy, jeśli wiele obliczeń wykonywanych jest w tej samej chwili czasu, zaś rozproszony, gdy obliczenia wykonywane są w różnych miejscach. Oś pionowa na rysunku 3.1 rozszerza przestrzeń modeli obliczeń o interakcję ze światem zewnętrznym, która czyni je niealgorytmicznymi.



Rys. 3.1. Przestrzeń modeli obliczeń

Interakcja zaś nie tylko umożliwia rozwiązanie problemów, dla których nie da się sformułować algorytmu, ale także może ułatwić rozwiązanie problemu, dla którego algorytm istnieje. Systemy interaktywne są bowiem oparte na naturalnych zachowaniach ludzkich, co czyni je praktyczniejszymi od zestawu reguł oferowanego przez nieinteraktywny algorytm.

Beaudouin-Lafon [BL08] nazywa „mitem o perfekcji" pogląd, że każdy proces da się wyrazić algorytmem, a zatem w pełni kontrolować i uzyskać poprawne rozwiązanie. Autor twierdzi, że możliwość interakcji z systemem obala ten mit, co jednak nie czyni komputerów mniej użytecznymi, a wręcz przeciwnie – zwiększa ich możliwości obliczeniowe. Szczególną formą interakcji jest współdziałanie człowieka z komputerem z tego względu, że człowiek nie zawsze zachowuje się w sposób racjonalny i przewidywalny, a mimo to od wieków potrafi rozwiązywać szereg problemów decyzyjnych, z którymi do dzisiaj nie radzi sobie żaden automat.

Beaudouin-Lafon uważa też, że stworzenie wygodnego w użyciu interfejsu nie wyczerpuje tego, co należy rozumieć jako interakcję człowieka z komputerem. W szczególności interfejs nie powinien narzucać mu dodatkowych ograniczeń w wykonywanej pracy, a także umożliwiać mu jak najlepsze wykorzystanie możliwości komputera i odwrotnie, umożliwiać komputerowi zwiększenie możliwości twórczych człowieka.

W związku z tym w rozprawie postuluję, aby wykorzystać dokument w systemie informatycznym zarówno jako jednostkę *informacji* jak i *interfejsu* – po pierwsze dlatego, że stanowi on naturalny interfejs dla człowieka i nie narzuca mu żadnych zbędnych struktur utrudniających bądź ograniczających jego pracę, i po drugie, jak to zostało opisane w rozdziale 1, dokument może pełnić aktywną rolę w procesie, umożliwiając interakcję człowieka z komputerem. Co więcej, dokument może posiadać pewną autonomię i automatyzować część zadań schematycznych, takich jak przestrzeganie procedur organizacyjnych, aby w ten sposób odciążyć człowieka i pozostawić mu więcej czasu na to, co najbardziej istotne w jego pracy, czyli podejmowanie decyzji.

3.1.1. Rozwiązanie prawidłowe a poprawne

W przypadku niealgorytmicznych problemów decyzyjnych nie da się sformułować kryterium oceny decyzji, a więc nie można mówić o rozwiązaniach *poprawnych* (optymalnych). Można natomiast stwierdzić zgodność procesu z konkretną obowiązującą procedurą postępowania (regułami organizacji) – wówczas rozwiązanie takiego procesu określa się jako *prawidłowe*. Biorąc pod uwagę wieloletnie doświadczenie w budowaniu struktury organizacji, udziałowcy systemu ufają, że rozwiązanie prawidłowe jest najlepszym, jakie można było znaleźć. Reasumując, różnica polega na tym, że w algorytmicznych procesach decyzyjnych ocenie podlega konkretna decyzja (wynik), zaś w niealgorytmicznych *proces* dochodzenia do decyzji.

W istniejących organizacjach opartych na wiedzy, np. medycznych zespołach diagnostycznych, nawet z perspektywy prawa dotyczącego zagadnień odpowiedzialności cywilnej lekarza, definiuje się błąd jako postawienie niewłaściwej diagnozy na skutek obiektywnie sprzecznego z zasadami obowiązującymi w medycynie postępowania lekarza [Rob07]. Tak więc błędem nie jest postawienie diagnozy, która skutkuje utratą zdrowia lub życia pacjenta, ale odstępstwo od procedury medycznej. Jeśli lekarze zrobili wszystko, co w konkretnej sytuacji nakazuje procedura, nie można określić diagnozy jako błędnej, nawet jeśli pacjent ponosi szkodę na skutek tej decyzji. W przyszłości mogą ujawnić się nowe fakty lub pojawić nowa wiedza, która wykaże niewłaściwość diagnozy i w konsekwencji doprowadzi do modyfikacji pro-

cedury, a przez to usprawnienia działania organizacji. Nie zmienia to jednak faktu zaklasyfikowania decyzji jako prawidłowej w chwili jej podejmowania.

Celem zespołów pracowników wiedzy przy rozwiązywaniu niealgorytmicznych problemów decyzyjnych jest więc uzyskanie i przyjęcie wyniku za prawidłowy ze względu na zastosowaną procedurę jego otrzymania, zamiast poszukiwania kryterium, na podstawie którego można by uznać je za poprawne [God09].

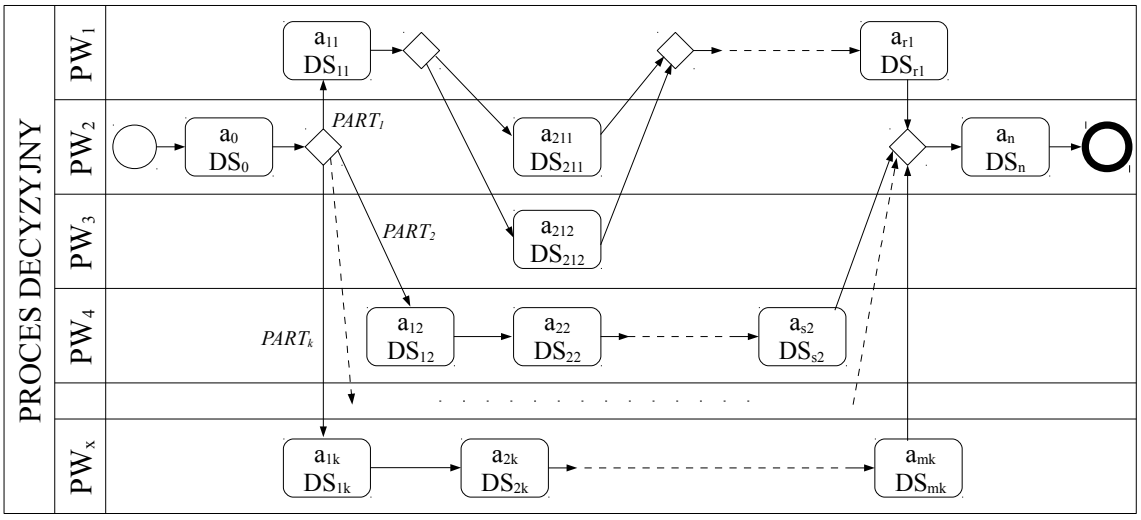
3.1.2. Podejmowanie decyzji w organizacjach opartych na wiedzy

Podejmowanie decyzji w organizacji opartej na wiedzy stanowi realizację modelu opisanego w podrozdziale 2.4, a więc procesu wiedzy łączącego w sobie wiedzę spersonalizowaną, ugruntowaną i skodyfikowaną. Procedura organizacyjna pozwala wybrać i zrealizować proces wiedzy regulujący postępowanie przy rozwiązywaniu danego problemu decyzyjnego. Proces ten ma ściśle określoną strukturę, zdefiniowaną przez model organizacji opartej na wiedzy (OOW) (podrozdział 2.4).

Uogólniony graf procesu wiedzy przedstawia rysunek 3.2. W opisie zostały użyte symbole graficzne *Business Process Model and Notation* (BPMN)[Obj11], służące do opisywania procesów biznesowych. Wizualnie, diagram ten przypomina *basen pływacki* (ang. *pool*) składający się z *torów pływackich* (ang. *lanes*). Każdy tor pływacki jest związany z jednym pracownikiem wiedzy i są w nim wyspecyfikowane czynności przypisane temu pracownikowi. Czynności *implementacyjne*, oznaczone prostokątami o zaokrąglonych wierzchołkach, wyznaczają miejsca w procesie, w których dany pracownik podejmuje decyzje na podstawie wiedzy zgromadzonej w dostarczonych mu dokumentach. Czynności zaznaczone jako okręgi wyznaczają początek i koniec procesu, zaś romby oznaczają czynności rozdzielające i łączące ścieżki przepływu dokumentów. Strzałki łączące czynności reprezentują tranzycje. Pełny zestaw elementów BPMN wykorzystywanych w rozprawie jest umieszczony na rysunku 5.3 (str. 103).

Istotną rolę w procesie mają pracownicy wiedzy PW, którzy w ramach konkretnej czynności a_i generują nową wiedzę w postaci pewnej decyzji składowej DS_i. Tak więc dokumenty wędrujące w sieci są zarówno nośnikami informacji, jak i elementami interakcji z pracownikami wiedzy, np. poprzez kontrolę zgodności treści z formatem, czy też kontrolę terminów, przy czym funkcjonalność dokumentów może być dowolnie rozszerzana. Początkowo zbiór decyzji składowych DS_0 ma postać pustych dokumentów określonego typu, które podczas procesu będą wypełniane wiedzą i informacją.

Proces rozpoczyna inicjator dokumentu (ang. *document originator*), który



Rys. 3.2. Graf przedstawiający pewien przykładowy proces wiedzy

konstruuje rdzeń dokumentu (zob. rys. 1.2, str. 19), a więc definiuje jego części – dokumenty składowe $PART_i$, strukturę grafu przepływu tych dokumentów oraz przyporządkowuje czynnościom pracowników wiedzy. Zgodnie ze strukturą przepływu pracy dane docierają do odpowiednich czynności, gdzie konkretni pracownicy podejmują decyzje składowe procesu DS_i. Decyzja składowa procesu jest rezultatem działania pracownika wiedzy na konkretnym dokumencie dostarczonym do węzła. Jest ona składową nowej wiedzy i informacji oraz powoduje przejście do kolejnej czynności zgodnie z określoną w grafie tranzycją.

Tranzycje określają rodzaje przejść między czynnościami. Przejścia mogą być bezwarunkowe – gdy po wykonaniu czynności i podjęciu decyzji składowej na danym węźle następuje przejście dokumentu do następnego węzła lub warunkowe – gdy po zrealizowaniu czynności dokonywany jest wybór kolejnych węzłów w zależności od pewnego warunku. Tranzycje mogą również dzielić (ang. *split*) lub łączyć (ang. *join*) ścieżkę przepływu dokumentów. W przypadku podziału dokument jest kopiowany lub dzielony na mniejsze części i wysyłany do każdego z wybranych węzłów. W czynności łączącej konieczne jest scalenie treści dokumentów i wówczas proces jest kontynuowany przez jeden scalony dokument.

Tranzycje warunkowe sprawiają, że nie ma konieczności, aby proces przeszedł przez wszystkie krawędzie grafu przepływu pracy. Również inicjator na początku może ustalić jakiś ogólny graf początkowy, który następnie może być rozszerzany zgodnie z założeniami procedury. Graf ma więc dynamicznie rozszerzalną strukturę. Mogą pojawiać się nowe przejścia, które nie były wcześniej planowane, a także nowe dokumenty.

Po przejściu przez określone czynności grafu przepływu pracy istnieje pewien

zbiór decyzji składowych DS_n, na podstawie którego pracownik wiedzy podejmuje decyzję kończącą proces. Wynik tej decyzji jest, według przyjętej w rozprawie terminologii, wynikiem prawidłowym.

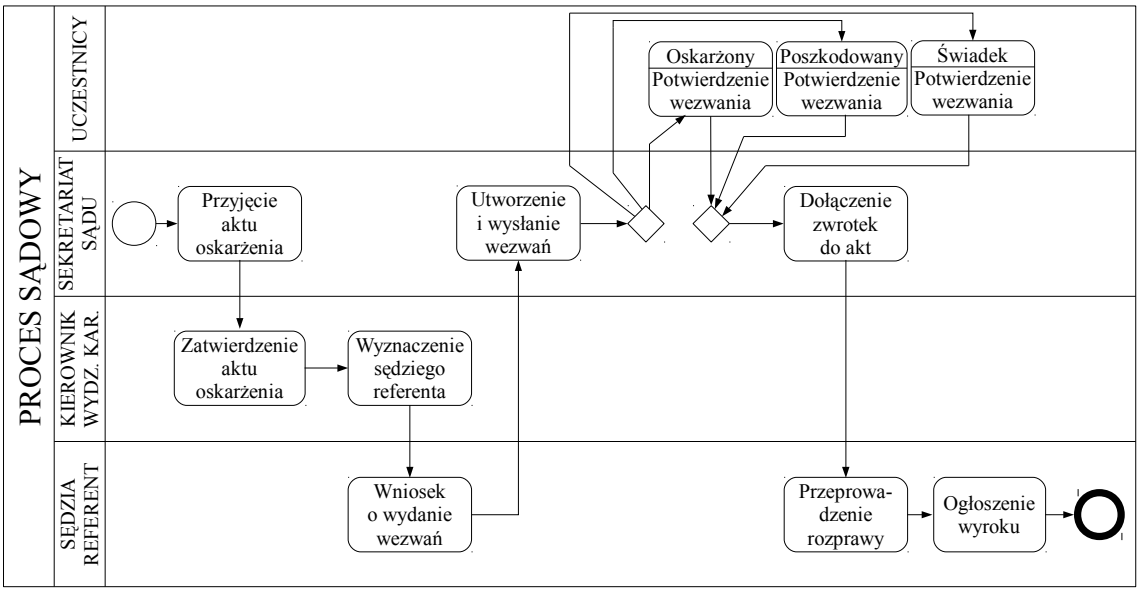
3.1.3. Przykłady organizacji realizujących niealgorytmiczne procesy decyzyjne

Rozwiązywanie niealgorytmicznych problemów decyzyjnych jest głównym zadaniem zespołów ekspertów zajmujących się nie tylko diagnozowaniem medycznym wspomnianym wcześniej, ale także badaniem przyczyn katastrof komunikacyjnych (szczególnie lotniczych), awarii wielkiej skali (np. elektrowni atomowych), lokalizacją i usuwaniem błędów w dużych systemach informatycznych, czy wreszcie wydawaniem wyroków w procesach sądowych. Tego typu problemy decyzyjne charakteryzują się dużym stopniem skomplikowania, dynamicznym napływem informacji, koniecznością wysuwania wniosków końcowych w warunkach niepełnej wiedzy i niekompletnej bądź też sprzecznej informacji.

Przykładowo, specjaliści w zakresie medycyny stają przed koniecznością zdiagnozowania pacjentów, u których symptomy chorobowe często są niejednoznaczne i nie wskazują wyraźnie jednostki chorobowej. Istnieje jednak określona procedura lekarska, która precyzuje postępowanie w konkretnych przypadkach i jest ona najczęściej opisana jako „dobra praktyka". Wśród lekarzy są różni specjaliści, z których każdy w ramach procedury podejmuje decyzję składową z własnego zakresu wiedzy. Na tę decyzję może wpływać opinia (decyzja składowa) innego specjalisty. Podczas procesu może dynamicznie zwiększać się również zasób informacji o stanie pacjenta. Efektem procesu jest konkretna decyzja, czyli diagnoza medyczna.

Procesem wiedzy jest również sprawa sądowa, przeprowadzana na podstawie bardzo szczegółowo opisanej procedury. Każda decyzja składowa musi być poparta dokumentem, którego zarówno format, jak i treść są ściśle zdefiniowane i niezbędne dla prawidłowego przebiegu procesu. Dla ilustracji posłużę się przykładem typowej sądowej sprawy karnej (rysunek 3.3), jednej z wielu rodzajów spraw rozpatrywanych przez sądy [Kan97]. Charakteryzuje ją duża liczba czynności przygotowawczych: wysyłanie wezwań, zatwierdzanie składu, wyznaczanie sędziego. Czynności procesowe to właściwa rozprawa sądowa, w której obieg dokumentów odbywa się w jednym miejscu, kończąca się ogłoszeniem wyroku.

Przedstawiony na rysunku 3.3 proces w sprawie karnej dla uproszczenia pomija czynności warunkowe, które często towarzyszą sprawom sądowym, takie jak np. odroczenie rozprawy, wybór innego wydziału sądu czy zmiana składu sędziowskiego, a także umieszcza uczestników (oskarżonego, poszkodowanego i świadka) na jednym



Rys. 3.3. Ogólny przepływ dokumentów w sprawie karnej z oskarżenia cywilnego

„torze pływackim" diagramu BPMN.

W procesach sądowych utworzone dokumenty zazwyczaj nie podlegają edycji. Jeżeli wymagają zmiany lub uzupełnienia tworzy się kolejne dokumenty (zwane wnioskami), które dokładnie opisują, co ma być zmienione lub uzupełnione w poprzednich dokumentach. Zatem praktycznie z każdą z przedstawionych na rysunku 3.3 czynności powiązany jest jeden lub więcej dokumentów, wchodzących w skład globalnego dokumentu zwanego aktami procesu. Są to w szczególności:

– *Przyjęcie aktu oskarżenia:* proces sądowy rozpoczyna się w momencie wpłynięcia aktu oskarżenia do sekretariatu sądu. Akt oskarżenia musi wpłynąć w tylu kopiach, ilu jest oskarżonych. Zostaje on następnie przekazany do kierownika odpowiedniego wydziału: w tym przypadku Wydziału Karnego.

– *Zatwierdzenie aktu oskarżenia:* kierownik wydziału zatwierdza akt oskarżenia. Może się zdarzyć sytuacja, że kierownik odsyła akt oskarżenia z powrotem do sekretariatu, jeśli nie spełnia on wymogów lub został skierowany do niewłaściwego wydziału.

– *Wyznaczenie Sędziego Referenta:* czynności tej towarzyszy zarządzenie o powołaniu składu sędziowskiego, wyznaczeniu czynności prawnych, terminu pierwszej rozprawy, zlecenie wezwań, wystąpienie o kartę karną.

– *Wniosek o wydanie wezwań:* czynności ta może być dodatkowo rozszerzona o powołanie biegłych, wnioski o przedłużenie aresztu tymczasowego itp. Oskarżeni oraz ich adwokaci otrzymują również kopię aktu oskarżenia.

– *Utworzenie i wysłanie wezwań:* odbywa się w sposób równoległy, aktualnie zazwyczaj listami poleconymi za pośrednictwem poczty.

– *Potwierdzenie przyjęcia wezwania:* jest równoznaczne z odbiorem listu poleconego i dostarczeniem do sekretariatu sądu tzw. zwrotki pocztowej, czyli potwierdzenia odbioru wezwania. Wezwanie powinno być dostarczone do uczestników postępowania w określonym odstępie czasu przed rozprawą.

– *Dołączenie zwrotek do akt:* zwrotki po dotarciu do sądu są dołączane do akt, jednakże nie wpływają na termin rozprawy, która się odbędzie w wyznaczonym wcześniej czasie. Mogą mieć natomiast wpływ na odroczenie rozprawy z przyczyn formalnych (w przypadku nieobecności któregoś z oskarżonych) bądź na jej przebieg (w przypadku nieobecności świadka). Osoba, która otrzymała wezwanie w odpowiednim terminie (co jest potwierdzone zwrotką), ma prawny obowiązek stawienia się na rozprawę. Zatem zwrotki mają istotny wpływ na postępowanie sądowe.

– *Przeprowadzenie rozprawy:* w trakcie rozprawy powstaje protokół oraz mogą się pojawić nowe dokumenty, dostarczone już na samej rozprawie przez strony, adwokatów czy biegłych. Ich włączenie do akt sprawy następuje na podstawie decyzji sędziego.

– *Ogłoszenie wyroku:* powstaje dokument wyroku (jego ewentualne uzasadnienie wydawane jest w innym trybie).

Dokumenty i procedury ich obiegu pełnią kluczową rolę w procesie sądowym. Powstaje wiele kopii tych samych dokumentów, np. aktu oskarżenia i zazwyczaj są one w formie drukowanej. Wąskim gardłem procesu jest przesyłanie dokumentów za pośrednictwem poczty. Jest to przyczyną wielu nieprawidłowości i nadużyć ze strony adresatów[1]. Jednakże dostarczenie pism sądowych pocztą elektroniczną jest często uważane przez sąd za nieważne, gdyż wymagany jest pocztowy dowód nadania. Nieprawidłowe wysłanie pism sądowych może natomiast skutkować nawet przegraniem sprawy przez stronę postępowania. Sąd może bowiem uznać, że takich pism w ogóle nie ma. Szczególnie rygorystyczne procedury pod tym względem są w przypadku spraw gospodarczych. Przeciąganie tych spraw lub nieuznanie jakiegoś pisma powoduje często poważne konsekwencje finansowe (przykładowe skutki takich działań zostały przedstawione w opartym na faktach filmie „Układ zamknięty" [Bug13]).

[1]Przykłady i opinie na temat przebiegu spraw sądowych i obiegu dokumentów opisuję na podstawie konsultacji z adwokatem mec. Tadeuszem Stolmannem z kancelarii adwokackiej w Kartuzach, jakie przeprowadziłam w dniu 04.04.2011r.

Od roku 2010 funkcjonuje w Polsce e-sąd, który rozpatruje najprostsze pozwy o zapłatę w trybie upominawczym z całej Polski. E-sąd formalnie jest VI Wydziałem Cywilnym Sądu Rejonowego Lublin-Zachód w Lublinie. Sąd ten jednak posiada swoje własne procedury, które stanowią znaczne uproszczenie procedur tradycyjnych sądów, a przez to stworzyły pole do nadużyć. Wykorzystują to nieuczciwe firmy windykacyjne i banki podczas windykacji przedawnionych długów. Wierzyciel występuje do e-sądu o nadanie nakazu zapłaty w trybie upominawczym i żeby dłużnik nie złożył sprzeciwu to wierzyciel celowo „myli się", podając fałszywy adres, na który ma zostać wysłany nakaz zapłaty. E-sąd zgodnie z uproszczoną procedurą nie weryfikuje poprawności danych odbiorcy i wysyła nakaz zapłaty na adres wskazany przez wierzyciela. Osoba mieszkająca pod tym nieprawidłowym adresem zazwyczaj wyrzuca mylnie zaadresowane awizo, zaś po dwukrotnym awizowaniu przesyłki, nadawca uważa ją za prawidłowo doręczoną ze wszystkimi konsekwencjami prawnymi. Od tego czasu dłużnik ma 14 dni na złożenie sprzeciwu od nakazu zapłaty, tyle że zazwyczaj nie wie, że ten nakaz otrzymał. Często dowiaduje się o tym dopiero od komornika [Obr12].

Przykład e-sądu pokazuje, jak eliminacja jednego z zasobów wiedzy organizacji (wiedza ugruntowana w procedurach organizacji, podrozdział 2.2), prowadzi do zaprzeczenia idei organizacji opartej na wiedzy. Pokazuje także, że receptą na przyśpieszenie procesów wiedzy nie jest uproszczenie procedur, a raczej wsparcie i automatyzacja przepływu pracy, tak by nadużycia, pomyłki lub opóźnienia nie miały miejsca podczas realizacji procesu wiedzy.

Prezentowana w rozprawie architektura MIND stanowi propozycję prawidłowej realizacji procesu wiedzy, poprzez jak najdalej posuniętą automatyzację przekazywania odpowiednich dokumentów bezpośrednio uczestnikom procesu oraz wykorzystywanie narzędzi znanych użytkownikom do tworzenia dokumentów. Należy jednak podkreślić, że w przypadku sądów wszelkie próby informatyzacji procedur wiążą się z problemami legislacyjnymi (potrzebne są odpowiednie ustawy, co wymaga czasu i rozwagi ustawodawcy we wdrażaniu nowych technologii) jaki i środowiskowymi (prawnicy obawiają się wysyłania pism drogą elektroniczną i opierają swoje taktyki na aktualnych zawiłościach proceduralnych).

3.2. Komunikacja i koordynacja czynności w procesach wiedzy

Procesy realizowane w organizacjach opartych na wiedzy, takich jak sądownictwo czy służba zdrowia, stanowią ugruntowaną wiedzę organizacji (przedstawione

w podrozdziale 2.2), która jest narzucona z zewnątrz przez nadrzędne organy lub wynika z doświadczenia i praktyk wypracowanych w organizacji. W praktyce proces wiedzy określa przepływ oraz typy dokumentów stanowiących interfejs dla pracowników. Skuteczne koordynowanie czynności w procesach wiedzy wiąże się zatem z możliwością organizacji wszystkich czynności procesu w logiczną strukturę przepływu pracy. Nasuwa się w naturalny sposób pytanie: jakie sytuacje związane z przepływem pracy mogą zaistnieć w organizacjach w związku z obiegiem dokumentów?

Wil van der Aalst i Arthur ter Hofstede podjęli w 1999 roku inicjatywę „Workflow Patterns Initiative", której zadaniem była szczegółowa analiza realnych organizacji, a także prototypów badawczych i standardów dotyczących modelowania procesów biznesowych w celu znalezienia kompletnego zbioru wzorców przepływu pracy. Punktem wyjścia do tych prac był początkowy zestaw 20 wzorców przepływu sterowania opisany w [AHKB03, Aal03]. Następnie rozwinęli ten zbiór, identyfikując 4 grupy wzorców różniących się perspektywami w procesach przepływu pracy. Grupy te obejmują: przepływ sterowania (ang. *control-flow*), dane (ang. *data*), zasoby (ang. *resource*) i obsługę wyjątków (ang. *exception handling*) [ARH11].

Celem autorów inicjatywy było zapewnienie teoretycznych i koncepcyjnych podstaw dla technologii realizujących wszelkie procesy biznesowe. Wykazali oni, że dowolnie rozbudowane procesy przepływu pracy składają się z elementarnych wzorców, które tworzą relatywnie niewielki i dobrze zdefiniowany zbiór. Zbiór ten może być wykorzystywany do oceny i porównywania możliwości istniejących języków lub produktów przepływu pracy oraz do tworzenia nowych technologii realizujących przepływ pracy [AHKB03, Aal03, HA05].

Z punktu widzenia rozprawy najbardziej interesująca jest, wymieniona przez autorów jako pierwsza, grupa wzorców przepływu sterowania, gdyż zbiór wzorców w niej zawarty związany jest z zagadnieniem koordynowania czynności wykonywanych przez pracowników wiedzy współpracujących w ramach organizacji [God10b]. Przepływ sterowania dotyczy ustalenia zależności pomiędzy różnymi czynnościami procesu, takich jak równoległość (wątek przepływu pracy jest rozbijany i czynności są wykonywane niezależnie w tym samym czasie), synchronizacja (kilka wątków jest scalanych w ramach jednej czynności), wybór (wątek przepływu pracy jest przekierowany na tranzycje wybrane na podstawie pewnych warunków), itd.

W dalszej części rozdziału poddaję analizie wzorce z tej grupy z perspektywy mechanizmów przesyłania wiadomości drogą poczty elektronicznej. W pierwszej kolejności przedstawię ogólne formy komunikacji w rozproszonych systemach, a następnie przeanalizuję wzorce przepływu sterowania w odniesieniu do dostępnych mechanizmów poczty elektronicznej. Ma to na celu redukcję aktualnego zbioru

wzorców przepływu sterowania [ARH11] oraz ukazanie ograniczeń poczty elektronicznej w koordynowaniu przepływu dokumentów. Przegląd ten stanowi też obszerne wprowadzenie do rozdziału 5. rozprawy, w którym zaprezentuję realizację wzorców przepływu sterowania w ramach architektury MIND.

3.2.1. Podstawowe formy komunikacji

Model organizacji opartej na wiedzy definiuje zbiory czynności, tranzycji oraz typów dokumentów. Jak już wspomniałam w podrozdziale 2.4, czynności i tranzycje są głównymi elementami przepływu pracy, zaś dokumenty stanowią interfejs użytkownika i są nośnikami informacji przekazywanej między pracownikami. W przypadku wykorzystywania w organizacji poczty elektronicznej do przekazywania dokumentów, czynnością można nazwać pewną pracę, którą pracownik wiedzy wykonuje po otrzymaniu wiadomości elektronicznej z jednym lub kilkoma załącznikami. W pierwszej kolejności dokumenty są wydobywane z wiadomości (zazwyczaj przez klienta pocztowego na podstawie przypisanych im typów MIME) i zapisywane w lokalnych katalogach odbiorcy, a następnie przetwarzane przy pomocy aplikacji komputerowych z wykorzystaniem dostępnych zasobów. Po zakończonej pracy aktualna wersja dokumentu jest załączana do nowej wiadomości elektronicznej i wysyłana do kolejnego pracownika wiedzy.

Tranzycją w przypadku przesyłania dokumentów drogą elektroniczną jest połączenie nadawca – odbiorca ustalane dynamicznie przez nadawcę w trakcie tworzenia wiadomości. W ogólności, komunikaty w systemach rozproszonych mogą być przesyłane w ramach jednego z trzech typów zdarzeń komunikacyjnych [KW98]:

- *jeden-do-jednego* (1-1), kiedy jeden nadawca wysyła pojedynczą wiadomość do jednego odbiorcy w jednej akcji,
- *jeden-do-wielu* (1-n), kiedy jeden nadawca wysyła pojedynczą wiadomość do wielu odbiorców w jednej akcji,
- *wielu-do-jednego* (n-1), kiedy wielu nadawców wysyła niezależnie od siebie, powiązane ze sobą wiadomości do jednego odbiorcy, wykonując równolegle swoje pojedyncze akcje,

Zdarzenie 1-1 w (rozproszonych) systemach poczty elektronicznej zachodzi, gdy jeden nadawca wysyła wiadomość na jeden znany mu adres mailowy odbiorcy, zaś zdarzenie 1-n zachodzi, gdy wiadomość jest wysyłana jednocześnie do wielu odbiorców. Adresy odbiorców mogą wówczas być podane bezpośrednio w formie listy w polach `To:` lub `Cc:`, pośrednio w postaci *aliasu* do wielu adresów

lub z wykorzystaniem adresu *listy mailingowej*. W pierwszym przypadku adresy odbiorców są znane nadawcy – takie zdarzenie komunikacyjne nazywa się *multicast*, w pozostałych dwóch przypadkach adresy odbiorców nie są znane nadawcy – takie zdarzenie określa się jako *broadcast*.

Zdarzenie komunikacyjne n-1 również może być rozważane w kontekście poczty elektronicznej, jednakże klient pocztowy zazwyczaj ma ograniczone możliwości wydobywania ze skrzynki odbiorczej wiadomości wysłanych od różnych nadawców i grupowania ich w zależności od pewnego globalnego kontekstu. Relacje pomiędzy wiadomościami można uzyskać poprzez nadawanie im przez nadawców unikalnych identyfikatorów w polu tematu wiadomości (`Subject:`). Dzięki temu odbiorca bądź jego klient pocztowy może przenosić oznaczone wiadomości do odpowiednich katalogów. Jest to jednak nieformalny sposób grupowania wiadomości i może powodować różnego typu błędy, np. nadawca może zapomnieć lub błędnie wpisać identyfikator przy tworzeniu wiadomości. Poza tym zbyt wiele katalogów w skrzynce odbiorczej może być przyczyną błędnego grupowania wiadomości przez odbiorcę.

Zdarzenie *wielu-do wielu* (n-n) jest generalnie trudne do implementacji w systemach rozproszonych [KW98] i trudno dla niego znaleźć pewien globalny kontekst w celu grupowania przesyłanych wiadomości. W przypadku systemów pocztowych, komunikacja n-n oznacza w praktyce wielokrotne wysyłanie i odbieranie maili bez żadnego porządku między członkami grupy pracującymi nad jakąś sprawą. Spontaniczne wysyłanie wiadomości elektronicznych pomiędzy współpracującymi osobami nie odpowiada jednak realizacji formalnych procedur przez organizacje oparte na wiedzy, które wymagają ustalenia pewnego następstwa zdarzeń. Komunikację n-n można spotkać w organizacjach w przypadku konieczności przeprowadzenia grupowych konsultacji miedzy pracownikami. Jednak na ogół nie odbywają się one za pośrednictwem poczty elektronicznej, a raczej z wykorzystaniem mechanizmów grup dyskusyjnych, forum internetowego lub blogów.

Jedną z istotnych cech charakteryzujących każdą formę komunikacji za pośrednictwem poczty elektronicznej jest jej asynchroniczność. Nadawca, wysyłając wiadomość elektroniczną, właściwie umieszcza ją na serwerze pocztowym adresata i po tej akcji może kontynuować swoją pracę, nie czekając, aż wiadomość dotrze do odbiorcy. Również odbieranie wiadomości nie wstrzymuje w żaden sposób pracy wykonywanej przez odbiorcę, o ile nie oczekuje on pewnej konkretnej wiadomości potrzebnej do kontynuowania pracy.

3.2.2. Wzorce przepływu sterowania

Podstawowe formy przesyłania komunikatów przedstawione powyżej opisują ogólne zdarzenia, jakie mogą wystąpić w procesie przepływu dokumentów w rozproszonym systemie. Szczegółowe przypadki różnych sytuacji związanych z komunikacją w procesach biznesowych i koordynacją czynności opisują wzorce przepływu sterowania. W tej części rozprawy przedstawię skrótowo wzorce przepływu sterowania na podstawie [ARH11], aby następnie przedyskutować możliwość ich implementacji w systemach poczty elektronicznej.

Podstawowe wzorce przepływu sterowania. Proces przepływu pracy określa kolejność występowania poszczególnych czynności. Czynności te mogą być wykonywane jedna po drugiej bądź równolegle, zaś przejścia między nimi mogą zależeć od pewnych warunków. Podstawowe wzorce przepływu sterowania (ang. *basic control flow patterns*) dotyczą elementarnych aspektów koordynowania procesu i wynikają bezpośrednio definicji tych pojęć zaproponowanych przez Workflow Management Coalition (WfMC)[WfM99]. Poniżej przedstawione są wzorce należące do tego zbioru wraz z odniesieniem do przesyłania dokumentów drogą poczty elektronicznej:

- **Sekwencja** (ang. *sequence*) występuje, gdy po zakończeniu jednej czynności, uaktywniana jest kolejna. W przypadku poczty elektronicznej oznacza to wysłanie pojedynczej wiadomości do dokładnie jednego odbiorcy. Nie musi być przy tym dokonywany żaden wybór, tzn. nie ma innych alternatywnych odbiorców, do których mogłaby trafić dana wiadomość.

- **Rozbicie równoległe** (ang. *Parallel Split*) to punkt w procesie, gdzie jeden wątek przepływu pracy zostaje rozbity na kilka równoległych wątków bez żadnych dodatkowych warunków. Następujące po danym punkcie czynności są uaktywniane równocześnie bez żadnego ustalonego porządku. Ta sytuacja odpowiada wysłaniu pojedynczej wiadomości przez jednego pracownika do pewnej określonej liczby odbiorców.

- **Synchronizacja** (ang. *Synchronization*) to punkt w procesie, do którego zbiega się kilka wątków i zostają one zsynchronizowane w jeden. Czynność łącząca oczekuje na wszystkie wątki w ramach jednego procesu, które mają zostać połączone i dopiero po ich połączeniu przepływ pracy może być kontynuowany. Wzorzec ten zazwyczaj jest odpowiedzią na *rozbicie równoległe*. W przypadku poczty elektronicznej jest to sytuacja, gdy jeden pracownik odbiera pewną liczbę wiadomości wysłanych przez różnych nadawców i nie wykonuje żadnej czynności z nimi powiązanej, dopóki nie otrzyma wszystkich oczekiwanych wiadomości.

– **Wyłączny wybór** (ang. *Exclusive Choice*) to punkt w procesie, gdzie wątek przepływu pracy zostaje przekierowany na dokładnie jedną z kilku możliwych tranzycji na podstawie analizy pewnych warunków związanych z tym punktem. Jest to sytuacja, gdy jeden nadawca wysyła pojedynczą wiadomość do dokładnie jednego pracownika wybranego ze zbioru możliwych odbiorców.

– **Proste łączenie** (ang. *Simple Merge*) występuje wtedy, gdy w danym punkcie schodzą się dwie tranzycje, które na pewno nie mogły być wybrane równolegle. Ma to miejsce w sytuacji, gdy przed tym punktem został wykonany *wyłączny wybór*. W rezultacie do miejscu *prostego łączenia* na pewno dotrze tylko jeden wątek i nie jest konieczne czekanie na kolejne ani nie jest potrzebna synchronizacja. Zatem w systemie poczty elektronicznej pracownik wiedzy oczekuje na pojedynczą wiadomość od jednego z kilku potencjalnych nadawców, zaś po jej odebraniu może od razu wykonać kolejną czynność.

Podstawowe wzorce przepływu sterowania w zasadzie wyrażają struktury synchronizacji i selekcji wprowadzone wcześniej przez imperatywne języki programowania i nie pozwalają na tworzenie bardziej zaawansowanych struktur wyboru ścieżki czy łączenia wątków przepływu pracy. Kolejne wzorce pogrupowane w kilka klas opisują bardziej złożone formy przepływu pracy, które pozwalają na koordynowanie czynności w wielu sytuacjach, które mogą się zdarzyć w organizacjach ludzkich.

Wzorce zaawansowanego rozgałęziania i synchronizacji (ang. *Advanced Branching and Synchronization Patterns*). Ta klasa wzorców rozszerza kwestię rozdzielania i łączenia wątków przepływu pracy, analizując sytuacje występujące w rzeczywistych procesach. Początkowo zbiór ten składał się z czterech wzorców [AHKB03], a następnie został rozszerzony o kolejne [RHAM06].

– **Wielokrotny wybór** (ang. *Multi-Choice*): to wzorzec reprezentujący punkt w procesie, w którym na podstawie pewnych warunków wybierany jest podzbiór dostępnych tranzycji; wątek przepływu jest wówczas rozbijany i przekazywany równolegle poprzez wybrane tranzycje do następnych czynności. Jest to ogólny przypadek dla dwóch wcześniejszych wzorców: *rozbicia równoległego*, gdzie wybierane są wszystkie dostępne tranzycje i *wyłącznego wyboru*, gdzie wybierana jest tylko jedna tranzycja. W przypadku poczty elektronicznej jest to sytuacja, w której jeden pracownik wysyła pojedynczą wiadomość do pewnej liczby współpracowników wybranych z większej grupy możliwych odbiorców.

– **Zsynchronizowane łączenie** (ang. *Synchronizing Merge*) to wzorzec podobny do *synchronizacji* z tą różnicą, że liczba oczekiwanych wątków może być

mniejsza niż liczba tranzycji wejściowych do czynności łączącej. *Zsynchronizowane łączenie* jest często poprzedzone *wielokrotnym wyborem*. Dla wiadomości elektronicznych jest to sytuacja, w której odbiorca otrzymuje wiadomości od pewnej liczby osób należących do większej grupy potencjalnych nadawców. Problemem odbiorcy jest więc ustalenie dokładnej liczby wiadomości, które powinny zostać odebrane przed przystąpieniem do kolejnej czynności. Z tego względu oryginalny wzorzec *zsynchronizowanego łączenia* został następnie podzielony na trzy przypadki: *strukturalny*, *lokalny* i *ogólny*. Różnią się one sposobem wyznaczania liczby wątków potrzebnych do realizacji tego wzorca [RHAM06].

– **Wielokrotne łączenie** (ang. *Multi-Merge*) różni się od poprzedniego wzorca tym, że zbiegające się w tym punkcie wątki nie podlegają synchronizacji. Każdy z wątków wymaga tej samej czynności, która jest wykonywana zaraz po jego dotarciu do punktu łączenia. Wzorzec ten jest przydatny, w przypadku gdy pracownik zlicza odebrane wiadomości i dla każdej z nich wykonuje tę samą czynność.

– **Dyskryminator** (ang. *Discriminator*) to wzorzec definiujący punkt w procesie, który akceptuje dokładnie jeden przychodzący wątek. Gdy on nadejdzie, uaktywnia dalsze czynności procesu, zaś sam czeka na pozostałe wątki, aby je zignorować. Dopiero, gdy je zignoruje, może być użyty jeszcze raz. Wersją ogólniejszą dyskryminatora jest wzorzec nazwany **Częściowym łączeniem** (ang. *Partial Join*), w którym do kontynuacji procesu oczekiwany jest nie jeden wątek, a pewien podzbiór k wątków z n możliwych, zaś pozostałe $(n-k)$ wątków jest ignorowanych. W przypadku poczty elektronicznej jest to sytuacja, w której pracownik czeka na wiadomości od konkretnej liczby nadawców z większej grupy, a po ich otrzymaniu może wykonać odpowiednią czynność. Wiadomości, które przyjdą od pozostałych pracowników zostaną natomiast zignorowane, np. poprzez usunięcie ze skrzynki odbiorczej.

  W zbiorze wzorców przepływu sterowania [ARH11] *dyskryminator* występuje w kilku wersjach. Opisana powyżej wersja jest nazwana strukturalną (ang. *structured discriminator*) i zakłada ignorowanie wątków w punkcie ich odbioru, a więc w przypadku poczty elektronicznej w skrzynce odbiorczej pracownika. Wersja anulująca (ang. *cancelling discriminator*) dyskryminatora zakłada anulowanie wątku w miejscu, w którym aktualnie się znajduje, co staje się dużo trudniejsze i kłopotliwe do realizacji w przypadku poczty elektronicznej, gdyż wiązałoby się z koniecznością wysyłania dodatkowych komunikatów w celu znalezienia aktualnej lokalizacji dokumentu. Dyskryminator blokujący

(ang. *blocking discriminator*) zaś uniemożliwia przesyłanie wątków należących do innej instancji procesu, dopóki nie zakończy ignorowania wszystkich wątków należących do bieżącej instancji. Wiąże się to z koniecznością zablokowania możliwości wysyłania wiadomości do danego pracownika, czego w systemach pocztowych się nie stosuje.

– **Uogólnione łączenie typu AND** (ang. *Generalized AND-Join*) to wzorzec podobny do *synchronizacji*, czyli kilka wątków jest przekształcanych w jeden z tą różnicą, że wątki mogą wpływać wielokrotnie przez daną tranzycję i za każdym razem wymagają synchronizacji z innymi wątkami w odpowiedniej kolejności, czyli wszystkie wątki, które wpłyną jako pierwsze są synchronizowane, następnie wszystkie, które wpłyną jako drugie itd. W przypadku komunikacji za pośrednictwem poczty elektronicznej odbiorca otrzymuje wiadomości od swoich współpracowników w sposób okresowy i dla każdego „kompletu" wiadomości wykonuje określoną czynność.

– **Rozdzielenie wątków** (ang. *Thread Split*) i **Łączenie wątków** (ang. *Thread Merge*) to wzorce, które pojedynczy wychodzący wątek aktywują określoną liczbę razy i przekazują poprzez jedną tranzycję (*rozdzielenie wątków*) lub łączą wszystkie przychodzące wątki z jednej tranzycji (*łączenie wątków*). Jest to sytuacja, w której jeden pracownik wysyła serię wiadomości do jednego współpracownika, a co za tym idzie, jeden pracownik odbiera serię wiadomości od jednego nadawcy. Przykładem może być rozbijanie bardzo dużych dokumentów na kilka wiadomości elektronicznych.

Wzorce zwielokrotnionego wykonywania czynności (ang. *Multiple Instances Patterns*). Ta grupa wzorców dotyczy sytuacji, w których pewna praca wymaga wielokrotnego i równoległego wykonania tej samej czynności. W przypadku komunikacji za pośrednictwem poczty elektronicznej wzorce te odpowiadają możliwości wysyłania pojedynczych wiadomości do grupy odbiorców identyfikowanych przez alias pocztowy lub nazwę listy mailingowej. Jest to komunikacja typu *jeden-do-wielu*, przy czym nadawca wybiera tylko jeden adres, zaś rzeczywiste adresy odbiorców są dla niego niedostępne, a więc nie wie dokładnie do kogo wysłał wiadomość oraz ile osób ją otrzyma.

Gdy wymagana jest synchronizacja po wielokrotnym wykonaniu czynności, ich rezultaty powinny być scalone w taki sposób, aby odbiorca był przekonany, że otrzymał wiadomość utworzoną przez pojedyncze wykonanie czynności. Zatem może to wymagać pewnego zewnętrznego mechanizmu, który zbierze wszystkie rezultaty wielokrotnie wykonanej czynności, scali je i wyśle jako pojedynczą

wiadomość z jednego adresu do wyznaczonego odbiorcy. Innym rozwiązaniem może być umieszczenie wszystkich w ten sposób powstałych wiadomości bezpośrednio w systemie plików odbiorcy i następnie wykonanie operacji scalenia. Wymagałoby to dodania dodatkowych możliwości do klientów pocztowych, np. do każdej wysyłanej wiadomości dołączany byłby odnośnik wskazujący odpowiedni serwis umożliwiający ich scalenie.

Wzorce zwielokrotnionego wykonania tej samej czynności zostały podzielone na dwie grupy:

- **Wzorzec nie wymagający synchronizacji** (ang. *Multiple Instances without Synchronization*) to wzorzec podobny do *wielokrotnego łączenia*, a więc każde wykonanie czynności odbywa się równolegle i jego rezultat jest rozpatrywany niezależnie od rezultatów innych wykonań.

- **Wzorce wymagające synchronizacji** (ang. *Multiple Instances with Synchronization*) to grupa wzorców, w których efekty zwielokrotnionego wykonania danej czynności są synchronizowane przed zwróceniem końcowego rezultatu. Wzorce w tej grupie różnią się sposobem wyznaczania liczby wykonań czynności wymaganych do synchronizacji. Liczba ta może być określana w trakcie projektowania przepływu pracy (ang. *a Priori Run-Time Design-Knowledge*), już po uruchomieniu przepływu, ale przed którymkolwiek wykonaniem czynności (ang. *with a Priori Run-Time Knowledge*) lub już po pewnej liczbie wykonań danej czynności (ang. *without a Priori Run-Time Knowledge*). Rezultaty zwielokrotnionego wykonania czynności mogą być też synchronizowane wybiórczo – pozostałe podlegają wówczas anulowaniu, podobnie jak w przypadku *dyskryminatora* (ang. *Partial Join for Multiple Instances*).

Wzorce zależne od stanu (ang. *State-based Patterns*) opisują sytuacje, w których przepływ jednego wątku procesu zależy od stanu przepływu innego wątku w bieżącym bądź nawet w innym, zewnętrznym procesie. Takie powiązania poszczególnych wątków nie są standardowe w komunikacji za pośrednictwem poczty elektronicznej, w której wysyłanie wiadomości pomiędzy użytkownikami jest niezależne od komunikacji innych użytkowników. W zasadzie jedyną możliwością sygnalizowania stanu pracy przez współpracownika jest wymiana dodatkowych komunikatów. Zatem aby zrealizować te wzorce w oparciu o pocztę elektroniczną, konieczne będą dodatkowe kanały komunikacyjne, takie jak połączenia telefoniczne czy komunikatory internetowe:

- **Odroczony wybór** (ang. *Deferred Choice*) to punkt w procesie, w którym wątek przepływu powinien być przekierowany na jedną z alternatywnych

tranzycji, jednak informacje uzależniające wybór tej tranzycji mogą być niedostępne w momencie dotarcia wątku do tego punktu, tj. tranzycje możliwe do wyboru są dostępne, ale sam wybór między nimi może być opóźniony do czasu pojawienia się pewnego zdarzenia. Jest to sytuacja w której pracownik nie wie, do kogo powinien wysłać wiadomość i czeka na dodatkową informację, która pomoże mu w dokonaniu odpowiedniego wyboru.

– **Przeplatany obieg równoległy** (ang. *Interleaved Parallel Routing*) to wzorzec określający zbiór czynności, które tworzą strukturę grafu z połączeniami równoległymi i sekwencyjnymi. Jeśli czynności są ustawione równolegle, mogą być wykonywane w dowolnej kolejności, zaś czynności ustawione sekwencyjnie powinny być wykonane w ustalonej przez sekwencję kolejności. Jednak kluczowym założeniem jest to, że żadne dwie czynności z tego zbioru nie mogą być aktywne w tym samym czasie, ale każda czynność powinna być wykonana. Wzorzec ten może odpowiadać sytuacji, w której wiadomość jest wysłana do grupy odbiorców, którzy fizycznie korzystają z tego samego urządzenia osobistego w celu sprawdzenia poczty elektronicznej.

– **Kamień milowy** (ang. *Milestone*) określa punkt w procesie, w którym pewna czynność została zakończona, zaś następująca po niej czynność jeszcze się nie rozpoczęła. W tym czasie mogą być wykonywane inne czynności, niezwiązane z tamtym przepływem. W przypadku poczty elektrowniczej może być to sytuacja, w której wiadomość została wysłana, zaś jej nadawca wykonuje inne, niezależne czynności, dopóki nie dostanie potwierdzenia, że jego wiadomość dotarła do odbiorcy.

Wzorce anulujące i kończące (ang. *Cancellation and Termination Patterns*) dotyczą różnych możliwości rozstrzygnięcia wątku lub całego procesu. Proces może zostać przerwany na skutek wystąpienia wyjątku spowodowanego pewnym zewnętrznym zdarzeniem (anulowanie) lub może się zatrzymać zgodnie z założeniami struktury procesu (zakończenie).

Anulowanie zazwyczaj wykracza poza możliwości klientów pocztowych, gdyż prawidłowo wysłana wiadomość nie może być usunięta przed dotarciem do skrzynki odbiorczej uczestnika wskazanego jako adresat. W procesie przepływu pracy można wyróżnić trzy typy wzorców anulujących:

– **Anulowanie czynności** (ang. *Cancel Activity*) oznacza zwykłe anulowanie aktywnej czynności przed lub w trakcie jej wykonywania.

– **Anulowanie sprawy** (ang. *Cancel Case*) to usunięcie całej instancji procesu, czyli wszystkich aktualnie aktywnych czynności oraz tych, które mogą być

wykonane w przyszłości, łącznie z ich podprocesami. Instancja procesu zostaje zapamiętana jako zakończona bez sukcesu.

– **Anulowanie części sprawy** (ang. *Cancel Region*) to możliwość anulowania zbioru czynności w ramach instancji jednego procesu. Jeśli jakieś czynności są aktualnie wykonywane, wówczas są wycofywane, przy czym czynności te nie muszą być połączonym podzbiorem w ogólnym modelu procesu.

Wprowadzenie dodatkowych kanałów komunikacyjnych, np. połączeń telefonicznych, jest często niewystarczające do realizacji wzorców anulujących w rozproszonych systemach wykorzystujących pocztę elektroniczną, z powodu braku mechanizmów kontroli stanu globalnego przepływu wiadomości w celu śledzenia aktualnych lokalizacji przesyłanych wiadomości. Oczywiście, można wysłać wiadomość „anulującą" do wszystkich potencjalnych współpracowników, jednak spowoduje to dostarczenie już często nieaktualnej wiadomości do wielu odbiorców, a zarazem może okazać się niewystarczające w przypadku dodania jakiegoś pracownika do procesu już po jego rozpoczęciu, o czym może nie wiedzieć nadawca wiadomości „anulującej".

Natomiast wzorce związane z planowanym zakończeniem procesu można podzielić na:

– **Niejawne zakończenie** (ang. *Implicit Termination*) występuje, gdy wszystkie wątki procesu zrealizują swój przepływ i nie mają już żadnej dalszej pracy do wykonania. Każdy wątek może kończyć się w innym czasie i zakończenie odbywa się w tym punkcie procesu, z którego nie ma już dla niego dalszej ścieżki przepływu. W przypadku komunikacji za pośrednictwem poczty elektronicznej jest to otrzymanie wiadomości, której nie trzeba już przekazywać dalej do nikogo.

– **Jawne zakończenie** (ang. *Explicit Termination*) występuje, gdy którykolwiek aktywny wątek dotrze do specjalnie wyróżnionej czynności kończącej. Wówczas wszystkie pozostałe wątki zostają anulowane analogicznie, jak w przypadku *anulowania sprawy*, tyle że instancja procesu zakończy się sukcesem.

Wzorce iteracyjne (ang. *Iteration Patterns*) określają sposób powtarzania pewnych czynności w procesie. Można wyróżnić następujące wzorce w tej klasie:

– **Dowolny cykl** (ang. *Arbitrary Cycle*) reprezentuje cykle niestrukturalne w procesie, które mają więcej niż jeden punkt wejścia lub wyjścia. Oznacza to możliwość wysłania wiadomości do współpracownika, który już ją wcześniej dostał, z dowolnego punktu procesu. Wzorzec ten może być zrealizowany

przez parę wzorców: *wyłączny wybór* i *proste łączenie* z klasy wzorców podstawowych.

– **Pętla strukturalna** (ang. *Structured Loop*) określa cykliczne wykonanie tej samej części procesu. Wzorzec definiuje pętle z pojedynczym punktem wejściowym i wyjściowym oraz z warunkiem ponownego wykonania rozpatrywanym na wejściu lub na wyjściu. W zasadzie jest to również połączenie wzorców: *wyłączny wybór* i *proste łączenie* z warunkiem rozpatrywanym w punkcie łączenia lub w punkcie wyboru.

– **Rekurencja** (ang. *Recursion*) zakłada ponowne uruchomienie tej samej czynności w trakcie jej wykonywania. Przykładem tego wzorca w systemie poczty elektronicznej może być wysłanie wiadomości przez nadawcę na swój własny adres. Nie jest to wcale dziwne zachowanie – wielu użytkowników wykorzystuje sporadycznie serwery pocztowe jako zewnętrzne nośniki danych do przenoszenia dokumentów pomiędzy swoimi urządzeniami w celu kontynuowania pracy w innym miejscu. Obecnie, wraz z rozpowszechnieniem się urządzeń mobilnych i usług synchronizacji systemów plików w oparciu o „chmurę" danych, ten sposób może wydawać się dosyć „toporny", niemniej jest nadal używany i całkiem skuteczny.

Wzorce wyzwalające (ang. *Trigger Patterns*) związane są z wysyłaniem i odbieraniem zewnętrznych sygnałów. Mogą one być wymagane do przepływu pracy w przypadku niektórych wzorców, np. wzorców zależnych od stanu czy anulujących. Nie można ich jednak zrealizować w oparciu o samą tylko pocztę elektroniczną, gdyż wymagają dodatkowych kanałów komunikacyjnych, wykraczających poza możliwości klientów pocztowych. Jednym z przykładów użycia zewnętrznych sygnałów jest dostarczenie wiadomości tekstowej SMS na telefon komórkowy zawierającej kod dostępu umożliwiający odczyt dokumentu dostarczonego drogą poczty elektronicznej.

3.2.3. Wzorce koordynujące czynności w systemach poczty elektronicznej

Opisane w podrozdziale 3.2.2 wzorce przepływu sterowania obejmują obecny stan wiedzy dotyczący modelowania procesów biznesowych oraz są niezależne od konkretnych technologii realizujących przepływ pracy. Stanowią one również klasę wzorców sprawdzoną w różnych aspektach użycia, od wielu lat wykorzystywanych do formalnego porównywania i oceny języków przepływu pracy [RHAM06, WRH+09]. Są brane również pod uwagę przez projektantów technologii workflow [Tog11a]. Z tego

względu w dalszej części rozprawy będę uznawać omówiony wcześniej zbiór wzorców za *kanoniczny*. W szczególności wykorzystam ten zbiór do zdefiniowania własnego zbioru wzorców przepływu mobilnych dokumentów interaktywnych i zaproponuję mechanizmy koordynowania czynności wykonywanych przez pracowników wiedzy w organizacjach sieciowych.

Tabela 3.2 przedstawia syntetyczne podsumowanie dotychczasowych rozważań. Kolumny NADAWCA i ODBIORCA opisują elementarne operacje, wykonywane przez pracowników wiedzy stosujących dany wzorzec w systemie pocztowy elektronicznej. Jak widać, oprócz wysyłania, odbierania i potwierdzenia odbioru wiadomości operacje te nie są bezpośrednio wspierane przez standardowych klientów pocztowych. Kolumna TYP określa typ zdarzenia komunikacyjnego (omówionego na str. 52), przy czym zapis ze znakiem „+", np. $(1\text{-}1)^+$, oznacza, że zdarzenie może wystąpić więcej niż jeden raz w danym wzorcu.

Postawmy zatem pytanie: czy zastosowanie koncepcji aktywnych dokumentów w systemie poczty elektronicznej pozwoli na skuteczną implementację zaprezentowanych wzorców bez konieczności ingerowania w standardowe protokoły pocztowe? Kolumna IMPL. tabeli 3.2 odpowiada na nie, na razie hipotetycznie, wskazując wzorce możliwe do zrealizowania z zastosowaniem proponowanego w rozprawie modelu mobilnego dokumentu interaktywnego (MIND). Sposób tej realizacji jest natomiast szczegółowo opisany w rozdziale 5. Wykażę, że implementacja tych wzorców pozwoli na koordynowanie czynności procesowych w organizacjach opartych na wiedzy, a więc umożliwi realizację procesów wiedzy.

Tabela 3.2: Implementowalność wzorców w systemach poczty elektronicznej

WZORZEC	SYSTEM POCZTY ELEKTRONICZNEJ			IMPL.
	TYP	NADAWCA	ODBIORCA	
1. Sekwencja	1-1	Jeden nadawca wysyła pojedynczą wiadomość	Jeden odbiorca otrzymuje jedną wiadomość	+
2. Rozbicie równoległe	1-n	Jeden nadawca wysyła wiadomość do grupy	Każda osoba z grupy odbiera wiadomość	+
3. Synchronizacja	n-1	Każdy członek grupy wysyła wiadomość	Jeden odbiorca oczekuje wiadomości od wszystkich nadawców	+
4. Wyłączny wybór	1-1	Nadawca wybiera jednego z kilku odbiorców	Dokładnie jeden odbiorca otrzymuje wiadomość	+
5. Proste łączenie	1-1	Jedna osoba z grupy nadawców wysyła wiadomość	Odbiorca oczekuje jednej wiadomości od kogokolwiek z grupy	+
6. Wielokrotny wybór	1-n	Nadawca wybiera podgrupę odbiorców	Wybrana grupa odbiorców otrzymuje wiadomość	+
7. Zsynchronizowane łączenie	n-1	Podgrupa nadawców wysyła wiadomość	Odbiorca czeka na wszystkie wiadomości od pewnej podgrupy	+
8. Wielokrotne łączenie	n-1	Grupa nadawców wysyła wiadomość	Odbiorca otrzymuje każdą wiadomość i działa na każdej z nich niezależnie	+
9. Dyskryminator	n-1	Każdy nadawca z grupy wysyła wiadomość	Odbiorca akceptuje pierwszą wiadomość a resztę odbiera i ignoruje	$\sim^a$
10. Uogólnione łączenie typu AND	$(n\text{-}1)^+$	Każdy nadawca wysyła przynajmniej jedną wiadomość	Odbiorca czeka na każdy kompletny zbiór wiadomości	+
11. Rozdzielanie/Łączenie wątków	$(1\text{-}1)^+$	Nadawca wysyła serię wiadomości	Odbiorca otrzymuje serię wiadomości	+
12. Zwielokrotnione wykonanie czynności	1-n	Nadawca wysyła wiadomość na pojedynczy adres związany z wieloma anonimowymi odbiorcami i oczekuje odpowiedzi	Odbiorca otrzymuje wiadomość	$\sim^b$
13. Odroczony wybór	1-1	Odbiorca czeka na adres właściwego odbiorcy	Odbiorca otrzymuje wiadomość	+
14. Przeplatany obieg równoległy	1-n	Nadawca wysyła wiadomość do grupy odbiorców	Każdy odbiorca otrzymuje wiadomość i w danym przedziale czasu ma ją na wyłączność	+
15. Kamień milowy	1-1	Nadawca może wysłać kolejny raz wiadomość dopóki nie otrzyma potwierdzenia odbioru	Odbiorca potwierdza odbiór pierwszej wiadomości	+
16. Anulowanie i Zakończenie	1-n	Nadawca identyfikuje grupę odbiorców, która ma anulować wiadomość	Odbiorca usuwa wiadomość ze swojej skrzynki	$\sim^b$
17. Przypadkowy cykl	$(1\text{-}1)^+$	Nadawca wysyła wiadomość do innego członka grupy, który mógł już ją otrzymać wcześniej	Odbiorca otrzymuje wiadomość	+
18. Strukturalna pętla	$(1\text{-}1)^+$	Nadawca wysyła wiadomość i oczekuje jej ponownie	Odbiorca otrzymuje wiadomość i przesyła ją dalej	+
19. Rekurencja	1-1	Nadawca wysyła wiadomość do siebie	Odbiorca otrzymuje wiadomość i odpowiada sobie samemu	+

Oznaczenia w kolumnie IMPL: „+” w pełni, $\sim$ tylko częściowo

[a] Strukturalny dyskryminator i częściowe łączenie [ARH11]

[b] Konieczna jest dodatkowa funkcjonalność serwera pocztowego

Rozdział 4

Model otwartej architektury mobilnych dokumentów interaktywnych (MIND)

Architektura dokumentu stanowi całościowy opis jego struktury oraz funkcjonalności. Struktura dokumentu definiuje jego poszczególne komponenty oraz powiązania między nimi, zaś funkcjonalność określa zachowania i polityki dokumentu realizowane podczas jego *cyklu życia*.

Proponowana w rozprawie architektura dokumentu MIND została zaprojektowana z myślą o realizacji procesów wiedzy przez organizacje w zorientowanym na dokumenty realistycznym trybie obliczeń zespołowych. Struktura dokumentu zakłada zatem możliwość definiowania dokumentów składowych dowolnego typu (określanego przez organizację) oraz ustalania kolejności i rodzaju czynności wykonywanych przez pracowników na danych dokumentach składowych. Funkcjonalność zapewnia natomiast koordynowanie wykonywanych czynności oraz interakcję dokumentów z użytkownikami.

Opracowana architektura MIND ma charakter otwarty, tzn. nie jest uzależniona od żadnej konkretnej technologii oraz daje możliwość rozbudowy zarówno struktury, jak i funkcjonalności. Pozwala to na dostosowanie jej do specyficznych wymogów danej organizacji.

Do zapisania modelu architektury mobilnych dokumentów interaktywnych (MIND) [Sic08, GW09, God12] został wykorzystany język XML i jego kilka standardowych notacji pokrewnych. Wybór języka XML był podyktowany jego popularnością, a także dostępnością narzędzi pozwalających na jego przetwarzanie i dodawanie funkcjonalności dokumentom opisanym za jego pomocą. Jako typ dokumentów tekstowych ze zdefiniowaną strukturą logiczną (rys. 1.1, str. 9), XML

jest zrozumiały dla człowieka oraz czytelny dla komputera.

Stwarza to szansę na przetrwanie języka XML w dłuższej perspektywie czasowej jako dominującego standardu reprezentacji dokumentów elektronicznych. Zapewnia on bowiem kompatybilność przyjętych rozwiązań z przyszłymi technologiami (ang. *forward compatibility*) oraz możliwości transformacji do innych notacji dokumentów (np. do „lżejszego" formatu JSON [Cro06]).

Punktem wyjścia dla architektury MIND było wykorzystanie dwóch popularnych mechanizmów:

1. automatycznego *wiązania danych* logicznych XML z obiektami (ang. *unmarshalling*), polegającego na przekształcaniu dokumentów XML na ich reprezentację obiektową [Bou11],

2. *mobilnych agentów*, czyli obiektów posiadających zdolność migracji między rozproszonymi lokalizacjami oraz autonomicznego wykonywania określonych usług [Fou02].

Mechanizmy te pozwalają na realizację cyklu życia dokumentu MIND (rys. 1.2, str. 19) szczegółowo opisanego w dalszej części tego rozdziału. Przedstawię w nim również metamodel oraz podstawowe komponenty opracowanej architektury MIND, warunkujące realizację cyklu życia dokumentu i implementujące model organizacji opartej na wiedzy opisany w wcześniej podrozdziale 2.4.

4.1. Cykl życia dokumentu MIND

Przedstawiony na rysunku 1.2. (str. 19) rdzeń dokumentu, który definiuje jego strukturę i zachowanie, jest zapisany w języku XML. Łączy on zasadnicze informacje zaczerpnięte z repozytorium dokumentów organizacji, wiąże poszczególne dokumenty składowe z pracownikami wiedzy, a także specyfikuje politykę migracji wynikającą z procedur organizacji. Wykorzystanie wspomnianego wcześniej mechanizmu wiązania danych pozwala na przekształcenie jednostek informacji zawartych w elementach dokumentu XML na obiekty w pamięci komputera (komponenty), które są następnie rozszerzane o funkcjonalność, zapewniającą im zdolność do migracji w otwartym systemie rozproszonym. Dzięki temu statyczna reprezentacja rdzenia dokumentu zostaje przekształcona na zbiór obiektów dynamicznych, zdolnych do samodzielnych działań na odległych komputerach.

Struktura dokumentu została zaprojektowana z myślą o realizacji procesów wiedzy, zawiera zatem w szczególności komponent specyfikujący ścieżkę migracji dokumentu oraz usługi (serwisy) umożliwiające interakcję migrujących obiektów

dynamicznych z pracownikami – głównymi wykonawcami czynności przypisanych dokumentom składowym – znajdującymi się na odległych stacjach roboczych, a struktura dokumentu jest skonstruowana z myślą o wsparciu pracowników w przestrzeganiu prawidłowego obiegu dokumentów.

W cyklu życia dokumentu MIND można wyróżnić trzy role przypisane jego użytkownikom:

- *projektant*, tworzy szablony dokumentów, dostosowując je do potrzeb danej organizacji, definiuje politykę migracji zgodną z procedurą oraz inne potrzebne polityki, a także wskazuje usługi niezbędne do realizacji procesu,
- *inicjator*, odszukuje w repozytorium odpowiedni szablon dokumentu i na jego podstawie tworzy rdzeń dokumentu dla konkretnego wykonania procesu: dostosowuje przepływ pracy, wyznacza konkretnych uczestników,
- *pracownik wiedzy*, jest uczestnikiem procesu przypisanym do konkretnych czynności.

Użytkownicy dynamicznych komponentów dokumentu pracują na swoich aktualnych stacjach roboczych, znajdujących się w terytorialnym rozproszeniu. Działają oni w charakterze pracowników wiedzy, pomiędzy którymi migrują dokumenty składowe, realizując procedurę organizacyjną i umożliwiając rozwiązanie niealgorytmicznego procesu decyzyjnego. Treść poszczególnych dokumentów składowych ulega wówczas rozszerzeniu, dodawane są kolejne jednostki informacji, a także są do nich dowiązywane dodatkowe treści w formie adnotacji. W trakcie realizacji procesu mogą być także dodawane nowe dokumenty składowe, a istniejące mogą być powielane, scalane lub usuwane, co powoduje, że liczba dynamicznych komponentów jest zmienna.

Migracja dynamicznych komponentów odbywa się tak długo, aż osiągną one swoje poszczególne cele, czyli zrealizują czynności opisane w ścieżce migracji. W końcowej fazie procesu zostają skierowane do ostatecznej lokalizacji, gdzie następuje ich scalenie i przekształcenie do statycznej postaci plików XML (ang. *marshalling*). Pliki te są archiwizowane w repozytorium do dalszego wykorzystania, w szczególności do ekstrakcji nagromadzonej w nim wiedzy lub konstruowania kolejnych procesów wiedzy.

Opracowana architektura MIND ma charakter otwarty, a zatem jej rozbudowa o kolejne komponenty w formie polityk pozwala na rozszerzenie funkcjonalności składników dokumentu. Do modelu dokumentu, w zależności od wymagań organizacji, można dołączyć kolejne mechanizmy, np. szyfrowania [SSW13], automatycznego negocjowania warunków wykonania czynności [Kac], podpisu cyfrowego, samodiagnozowania dokumentu czy metod odtwarzania jego stanu w wypadku awarii.

4.2. Metamodel dokumentu MIND

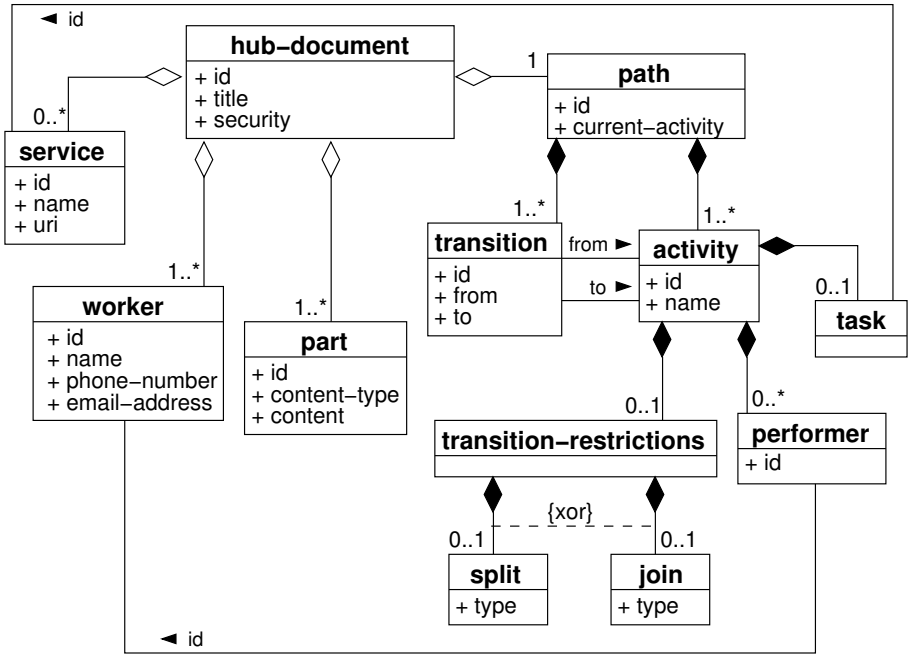
Głównym celem architektury MIND jest umożliwienie realizacji modelu organizacji opartej na wiedzy (przedstawionego w podrozdziale 2.4) poprzez przekształcenie statycznego dokumentu na zbiór mobilnych komponentów, które realizują proces wiedzy organizacji w środowisku rozproszonym. W procesie wiedzy istotne są: procedury organizacyjne oraz pracownicy wiedzy, którzy komunikują się za pośrednictwem dokumentów określonego typu.

Każdy dokument MIND składa się z zestawu plików, które są przekazywane poprzez sieć (np. drogą poczty elektronicznej) jako jeden *pakiet MIND*. Pakiet ten stanowi *statyczną* postać dokumentu MIND i zawiera zarówno pliki z treścią, jak i pliki będące zasobami dokumentu. Pliki zawierające treść to w rzeczywistości dokumenty dowolnego typu, na których pracownicy wiedzy wykonują swoją pracę. Zasobami dokumentu są zaś pliki definiujące politykę migracji oraz zawierające informacje na temat wydzielonych części dokumentu, pracowników i usług potrzebnych do realizacji procesu.

Pakiet MIND, po dotarciu na odpowiednią (zdefiniowaną w polityce migracji) lokalizację, jest rozpakowywany i umieszczany odpowiedniej strukturze katalogów na lokalnym urządzeniu pracownika. Pliki zasobów są następnie wiązane (ang. *unmashalling*) z wykonywalnymi obiektami – *komponentami dokumentu MIND*. Zbiór tych komponentów stanowi *dynamiczną* postać dokumentu MIND. Struktura katalogów i zawartych w nich plików z treścią oraz komponentów dokumentu MIND implementuje *rdzeń dokumentu* (ang. *hub-document*). Struktura rdzenia dokumentu (`hub-document`) jest przedstawiona na rysunku 4.1.

Obiekty usług (`service`) pozwalają na wyposażenie dokumentu w pożądaną funkcjonalność, która zmienia jego rolę w procesie na aktywną (zob. podrozdział 1.2). Dokument MIND zakłada możliwość występowania trzech rodzajów usług. Są to usługi:

- *wbudowane*, implementowane za pomocą skryptów lub fragmentów kodu przenoszonych przez obiekty i możliwe do wykonania bezpośrednio po przybyciu na wybraną lokalizację; dzięki temu obiekty mogą zrealizować swoją funkcjonalność niezależnie od lokalnych środowisk, które odwiedzają, realizując przypisaną ścieżkę migracji,
- *lokalne*, zaimplementowane jako skrypty, które sprawdzają dostępność i aktywują bądź zgłaszają zapotrzebowanie na narzędzia zainstalowane lokalnie na urządzeniu użytkownika,
- *zdalne*, specjalistyczne usługi dostępne na zewnętrznych serwerach, na które



Rys. 4.1. Schemat architektury dokumentu MIND w postaci dynamicznej

dynamiczny obiekt może zadeklarować zapotrzebowanie i z którymi łączy się w sposób zdalny lub wymaga ich instalacji na lokalnym komputerze.

Obiekty pracowników wiedzy (`worker`) dostarczają niezbędnych informacji dotyczących pracowników, aby mogli stać się wykonawcami przypisanych im czynności w procesie. W szczególności zawierają identyfikator (`id`) oraz adres poczty elektronicznej (`email-address`), które umożliwiają dostarczanie im pakietu MIND.

Obiekty dokumentów składowych (`part`) opisują i przechowują w sobie statyczne składniki dokumentu z treścią. Zbiór dokumentów składowych odpowiada zbiorowi D w modelu organizacji opartej na wiedzy (podrozdział 2.4). Każdy element tego zbioru ma przyporządkowany pewien typ zawartości (`content-type`) ze zbioru dostępnych typów S występującym we wzorze (2.2), np. typów MIME lub typów charakterystycznych dla danej organizacji.

Komponent ścieżki migracji (`path`) składa się z obiektowej reprezentacji przepływu pracy opisującego procedurę organizacyjną. W pakiecie MIND komponent ścieżki jest reprezentowany przez plik w formacie XPDL (dialekcie języka XML) służący do opisu przepływu pracy [WfM12b], zaś obiekty komponentu powstają na skutek wiązania standardowych elementów XPDL używanych w modelu architektury MIND. Obiekty czynności (`activity`), w zależności od typu, mogą zawierać informacje o swoich wykonawcach (`performer`), którymi są pracownicy wiedzy oraz usługach (`task`) uruchamianych w celu przetwarzania treści dostarczonych dokumentów. Zbiór obiektów czynności odpowiada zbiorowi A występującemu we wzorze (2.1),

opisującym model organizacji opartej na wiedzy. Obiekty tranzycji (`transition`) określają obieg dokumentów między czynnościami. Odpowiadają zbiorowi T występującemu we wzorze (2.1) opisującym model organizacji.

Koncepcja pakietu MIND łączącego w jednej strukturze dokumenty różnych typów przedstawiona na rysunku 4.1 przypomina do pewnego stopnia model zawartości SCORM (ang. *Sharable Content Object Reference Model*) [Adv09], opracowany w celu wymiany treści pomiędzy sieciowymi systemami zdalnego nauczania a narzędziami do tworzenia modułów e-learningowych. Pakiet SCORM łączy w sobie różnego typu zawartość oraz jej opis w postaci oddzielnego dokumentu XML zgodnego ze standardem SCORM. Dzięki temu zawartość jest podzielona na części, którym dodatkowo można przypisać ustaloną kolejność (strategię) wyświetlania. Struktura zasobów SCORM jest dobrze określona, ale format zawartości nie jest ustandaryzowany, co może powodować problemy właściwego wyboru aplikacji do jej przetwarzania (problemy te zostały szczegółowo opisane w podrozdziale 1.1.2). Dodanie do dokumentu funkcjonalności w formie usług pozwala na wyeliminowanie problemu prawidłowego odtwarzania treści bez konieczności uzależniania modelu od jakiegoś konkretnego typu zawartości. Model MIND nie stanowi rozszerzenia czy udoskonalenia modelu SCORM, a jedynie wykorzystuje tę samą koncepcję rozdzielenia treści od zasobów, które ją opisują.

4.3. Opis komponentów dokumentu MIND

Rdzeń dokumentu, który jest tworzony przez *inicjatora* z dostępnych szablonów (por. rys. 1.2) jest zasadniczo pusty, tj. nie zawiera jeszcze żadnych treści – ma natomiast strukturę gotową do przekształcenia go na zbiór obiektów dynamicznych, zdolnych do migracji i wykonywania samodzielnych działań w otwartym systemie rozproszonym.

Rdzeń tworzy strukturę katalogów na dysku. Katalog nadrzędny, o nazwie będącej identyfikatorem rdzenia, zawiera w sobie następujące pliki:

- `document.xml` z podstawowymi informacjami na temat dokumentu,
- `services.xml` ze specyfikacją usług przypisanych do dokumentu,
- `workers.xml` z danymi na temat pracowników wiedzy,
- `parts.xml` definiujący dokumenty składowe dokumentu MIND,
- `path.xml` będący dokumentem XPDL, definiujący ścieżkę przepływu pracy dokumentów składowych i zawierający informacje o wykonawcach danych czynności i usługach,

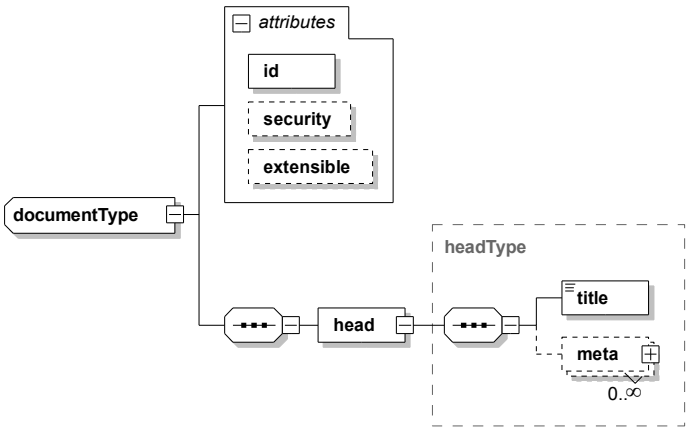
i katalogi:

- `Parts` z dokumentami określonego typu, wyodrębnionymi z pliku parts.xml w celu dalszego przetwarzania ich treści,
- `Scripts` z kodem usług wbudowanych, przyniesionych razem z dokumentem,
- `Artifacts` z plikami lokalnymi użytkownika, np. wersje tymczasowe dokumentów, własne pliki związane z dokumentem, które nie będą dalej przenoszone przez dokument.

Kolejne podrozdziały zawierają dokładny opis komponentów MIND. Opisujące je schematy XSD (XMLSchema), dla lepszej czytelności, zostały zaprezentowane w formie graficznych diagramów (rys. 4.2 – 4.15, z wyjątkiem rys. 4.11), wygenerowanych w programie Altova XMLSpy [Alt13]. Pełnotekstowa zawartość schematów XSD opracowanych dla komponentów MIND jest dołączona do rozprawy w załączniku A, zaś schemat XSD dla komponentu przepływu pracy (`path.xml`) jest standardowym opisem języka XPDL v.2.2 [WfM12a].

4.3.1. Komponent główny

Elementy i atrybuty komponentu głównego (`document.xml`) są przedstawione na rysunku 4.2. Atrybuty komponentu opisują tożsamość i podstawowe właściwości każdego dokumentu MIND. W chwili tworzenia rdzenia dokumentu przez inicjatora jest mu nadawany unikatowy identyfikator dokumentu `id`, zapisany jako ciąg znaków, np. `id = "HUB00001"`.



Rys. 4.2. Struktura logiczna głównego komponentu MIND

Wartości pozostałych atrybutów obejmują następujące parametry:

– `security`, określa poziom bezpieczeństwa dokumentu i przyjmuje jedną z wartości: `none`, `low`, `medium`, `high` lub `paranoid`[1]. Domyślna wartość atrybutu to `none`, zaś polityka bezpieczeństwa dokumentu specyfikująca jego zachowanie w zależności od innych wartości parametru może stanowić rozwinięcie architektury MIND i nie została opisana w rozprawie.

– `extensible`, określa możliwość dodawania nowych obiektów dynamicznych w trakcie cyklu życia dokumentu, skutkującego pojawieniem się nowych dokumentów składowych w dokumencie zakończonym. Atrybut przyjmuje wartość `true` lub `false`.

Element `<head>` komponentu głównego stanowi jego *nagłówek* i zawiera tytuł dokumentu oraz kolekcję elementów `<meta>`, zawierających informację przydatną przy przeszukiwaniu repozytoriów przechowujących dokumenty finalne w postaci statycznej. Informacja ta może występować pod trzema postaciami:

– informacji standardowej tylko do odczytu (`name="Generator"`), określającej nazwę narzędzia za pomocą którego wytworzono dokument,

– edytowalnej informacji standardowej, określającej podstawowe jednostki informacji bibliotecznej dotyczącej dokumentu (`Version`, `Keywords`, `Autor`, `Description`),

– edytowalnej informacji definiowanej przez inicjatora za pomocą własnych nazw (jako wartości atrybutu `name`), określającej np. status dokumentu, stan zaawansowania prac czy przewidywany czas zakończenia.

Przykładowa zawartość elementu `<head>` komponentu głównego dla czterech pierwszych rozdziałów tej rozprawy może mieć następującą postać:

```
<head>
  <title>Model otwartej architektury rozproszonych dokumentów
  elektronicznych umożliwiającej podejmowanie decyzji w trybie
  obliczeń zespołowych</title>
  <!-- informacja inicjatora: -->
  <meta name="Description"
       content="Rozprawa doktorska opisująca architekturę MIND" />
  <meta name="Keywords"
```

[1]Zestaw nazw został zapożyczony z instalatora systemu operacyjnego Linux, w którym służy do oznaczenia stopnia zabezpieczenia komputera przed atakami sieciowymi. Jest także powszechnie przyjęty do określania stopnia zabezpieczenia przeglądarek internetowych przed plikami cookie i skryptami wykonywalnymi.

```
        content="Rozprawa, MIND, XML, workflow, dokument" />
    <meta name="Author"
        content="Magdalena Godlewska" />
    <!-- informacja systemowa: -->
    <meta name="Version"
        content="1.0" />
    <meta name="Generator"
        content="TeXnicCenter" />
    <!-- informacja dodatkowa: -->
    <meta name="Custom-boolean-roboczy"
        content="true" />
    <meta name="Custom-number-procent"
        content="55" />
    <meta name="Custom-date-deadline"
        content="2013/06/10" />
</head>
```

W powyższym opisie dokumentu znalazły się informacje standardowe dotyczące inicjatora, numeru wersji i narzędzia jakie zostało użyte do generacji rdzenia, a także dodatkowe informacje przeznaczone do wiadomości dla pozostałych pracowników wiedzy (np. promotora rozprawy).

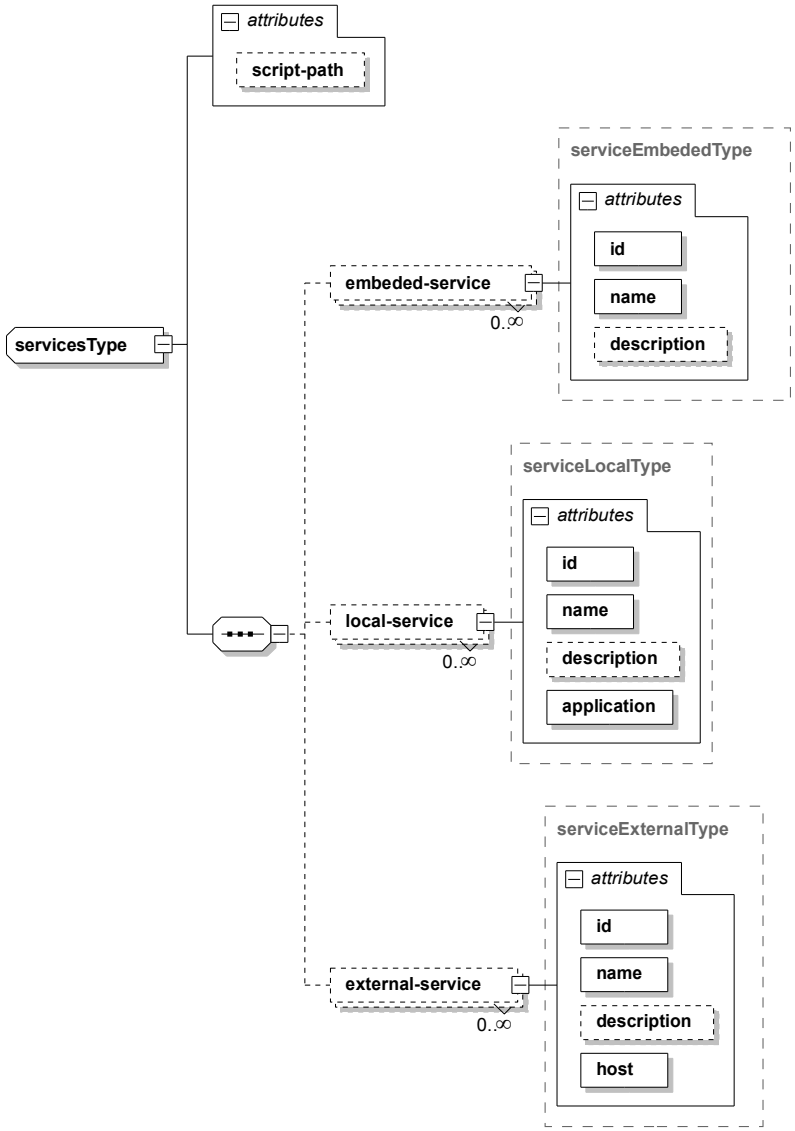
4.3.2. Komponent usług

Zawartość elementu `<services>` komponentu `services.xml` specyfikuje usługi dokumentu MIND, jakie są dostępne dla każdego obiektu dynamicznego utworzonego w wyniku konwersji z jego postaci statycznej do postaci obiektowej. Struktura logiczna tego elementu jest przedstawiona na rysunku 4.3.

Jak wspomniałam w podrozdziale 4.2, architektura MIND przewiduje możliwość deklaracji trzech rodzajów usług: wbudowanych, lokalnych i zdalnych. Odpowiadają im następujące elementy w schemacie usług:

- `embedded_service` (usługi wbudowane), opisane w tym elemencie usługi są przynoszone razem z dokumentem i umieszczane w jego strukturze katalogów na urządzeniu roboczym; wykonywane są bezpośrednio przez aplikację kliencką MIND w ramach czynności procesu i nie wymagają żadnych dodatkowych narzędzi do ich uruchamiania; do tej grupy usług należą przede wszystkim skrypty związane z przetwarzaniem struktury danego dokumentu, np. transformacje,

- `local_service` (usługi lokalne), opisują oczekiwane przez dokument lokalne



Rys. 4.3. Struktura logiczna komponentu usług

aplikacje zainstalowane na urządzeniu pracownika, które czynność procesu potrafi uruchomić lub zgłosić na nie zapotrzebowanie;

– `external_service` (usługi zdalne), opisują zewnętrznie dostępne usługi, np. serwisy sieciowe, do których czynność procesu ma zdalny dostęp.

Każda usługa identyfikowana jest za pomocą unikatowej wartości atrybutu `id` i posiada nazwę podaną jako wartość atrybutu `name` typu wyliczeniowego oraz opis w postaci łańcucha znaków, zapisanego jako wartość atrybutu `description`. Komponent usług dokumentu MIND `<services>` może posiadać atrybut specyfikujący ścieżkę dostępu do skryptów dostarczanych przez dokument.

Element `<local-service>` ma dodatkowo atrybut `application`, zawierający nazwę systemową aplikacji realizującej usługę wyspecyfikowaną atrybutem `name`. Jeśli wartość ta jest równa `default`, wówczas uruchamiana jest domyśla aplikacja systemowa danego typu, np. domyślna przeglądarka internetowa. Z kolei dodatkowy atrybut elementu `<external-service>`, specyfikującego usługi zdalne to `host`, zawierający adres URL usługi wyspecyfikowanej atrybutem `name`.

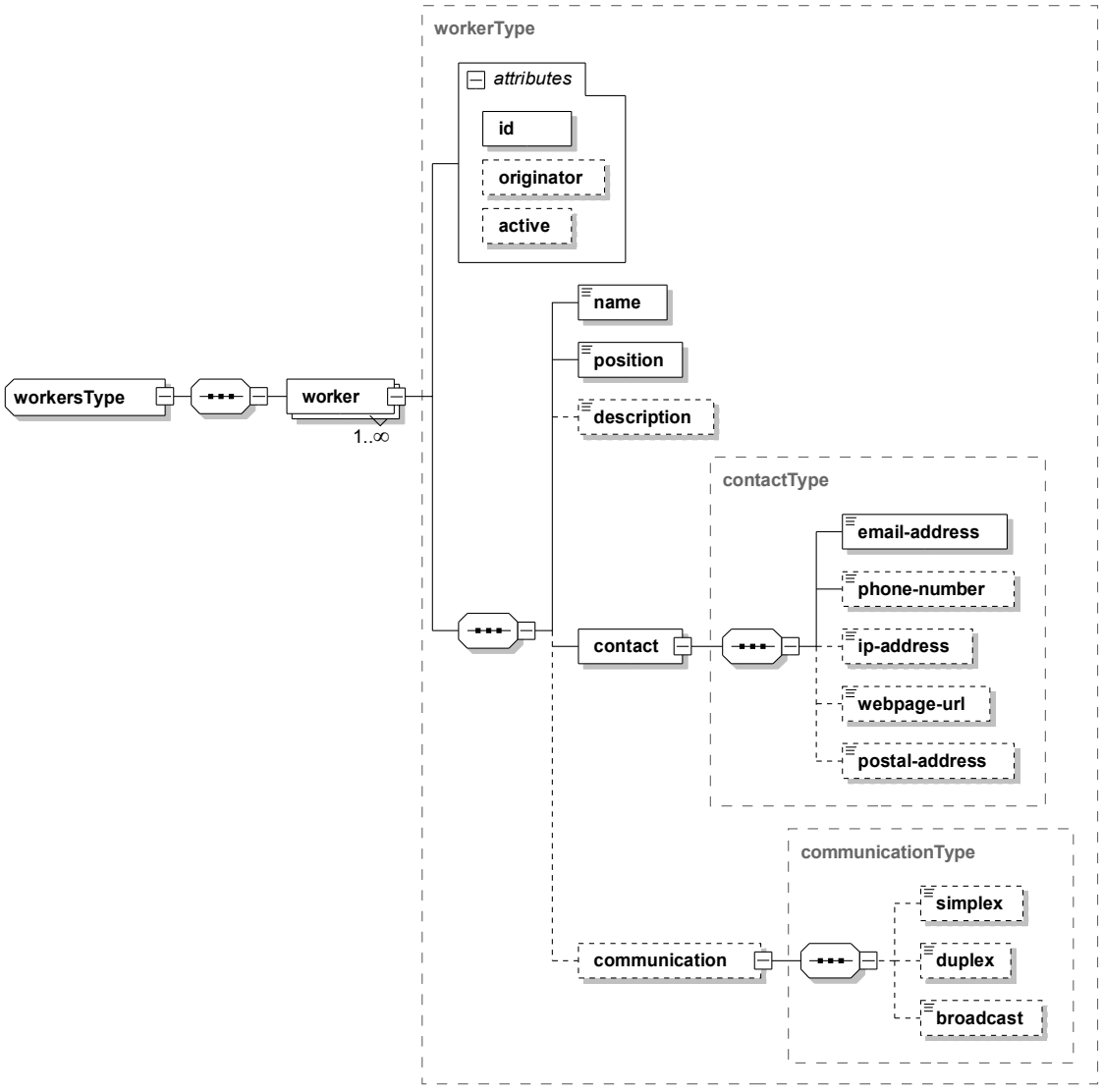
Przykładowa zawartość komponentu usług dla tej rozprawy może mieć postać:

```
<services script-path="/HUB00001/Scripts">
  <embedded-service id="SER00001" name="splitParts"
        description="Transformacja rozdzielająca rozdziały." />
  <embedded-service id="SER00002" name="mergeParts"
        description="Transformacja łącząca rozdziały." />
  <embedded-service id="SER00003" name="alarm"
        description="Kontrola upływu czasu." />
  <local-service id="SER00004" name="browser"
        description="WWW client" application="default" />
  <local-service id="SER00005" name="graphics editor"
        description="" application="LibreOffice Draw" />
  <local-service id="SER00006" name="text editor"
        description="" application="TeXnicCenter" />
  <local-service id="SER00007" name="PDF reader"
        description="" application="Adobe Reader" />
  <external-service id="SER00008" name="content converter"
        description="Konwerter typów dokumentów"
        host="http://menaid.org.pl/serwisy/content-converter.wsdl" />
  <external-service id="SER00009" name="image converter"
        description="Serwis sieciowy do konwersji grafiki"
        host="http://menaid.org.pl/serwisy/image-converter.cgi" />
</services>
```

Powyższy opis wskazuje z jakiej ścieżki mogą być uruchamiane skrypty dystrybuowane wraz z dokumentem. Specyfikuje trzy specjalizowane usługi wbudowane (`SER00001` – rozdzielająca części dokumentu, `SER00002` – łącząca części dokumentu oraz `SER00003` – monitorująca czas przetwarzania), cztery usługi systemowe udostępniane przez lokalny system operacyjny danej stacji (`SER00004` – przeglądarka internetowa, `SER00005` – program graficzny, `SER00006` – edytor tekstowy i `SER00007` – program do przeglądania dokumentów PDF) oraz dwie usługi zewnętrzne serwera dokumentów MIND (`SER00008` – konwertująca formaty cyfrowych dokumentów oraz `SER00009` – konwertująca obrazy cyfrowe).

4.3.3. Komponent pracowników wiedzy

Każdy dokument MIND ma w komponencie pracowników wiedzy (`workers.xml`) przypisanego co najmniej jednego pracownika, którym domyślnie jest inicjator tworzący dokument rdzeń na początku cyklu życia (por. rys. 1.2). Po konwersji statycznego dokumentu na dynamiczne obiekty i w trakcie ich migracji w systemie, mogą być dołączani kolejni pracownicy wiedzy. Atrybuty i zawartość elementu `<worker>` zostały przedstawione na rysunku 4.4.



Rys. 4.4. Struktura logiczna komponentu pracowników wiedzy

Pracownik wiedzy identyfikowany jest za pomocą unikatowej wartości atrybutu `id`, zaś inicjator dokumentu wyróżniany jest wartością atrybutu `originator="yes"`. Wartość atrybutu `active="yes"|"no"` wskazuje na rolę pracownika w organizacji lub w danym procesie: `"yes"` oznacza czynną rolę – pracownik bierze udział w two-

rzeniu treści dokumentów, `"no"` oznacza bierną rolę – w ten sposób można oznaczyć osoby otrzymujące dokumenty tylko do wglądu, w tym osoby niezatrudnione w organizacji[2]. Atrybut `active` określa też domyślne uprawnienia pracowników dotyczące dostępu do dokumentu: czynni pracownicy mogą modyfikować treści dokumentu, zaś bierni mogą jedynie przeglądać jego treść. Dodatkowe uprawnienia dostępu są definiowane odrębnie dla każdej części dokumentu i określane są w komponencie dokumentów składowych (zob. podrozdział 4.3.4).

Elementy `<name>` i `<position>`, podrzędne względem elementu `<worker>`, wyróżniają łańcuchy znaków specyfikujące odpowiednio imię i nazwisko pracownika oraz stanowisko (pełnioną funkcję) w organizacji. Opcjonalny element podrzędny `<description>` umożliwia zapisanie dodatkowych informacji dotyczących pracownika i jego funkcji w procesie. Element `<contact>` zawiera podstawowe dane niezbędne do zapewnienia kontaktu z danym pracownikiem, zapisywane w postaci tekstowej w elementach `<email-address>`, `<phone-number>`, `<ip-address>`, `<webpage-url>` i `<postal-address>`. Jeśli warstwą transportową dla komponentów dokumentu jest poczta elektroniczna, wówczas najistotniejszy jest element `<email-address>`, który musi mieć przypisaną wartość (linia ciągła wokół elementu na rysunku 4.4), pozostałe dane kontaktowe mogą występować opcjonalnie (linia przerywana). Wreszcie element podrzędny `<communications>` specyfikuje w postaci tekstowej inne informacje niezbędne do komunikowania się pracownika wiedzy pracującego z dokumentem składowym MIND, przebywającym aktualnie na jego lokalnej stacji roboczej.

Przykładowa zawartość elementów `<worker>` dla tej rozprawy może mieć następującą postać:

```
<worker id="PW00001" originator="yes" active="yes">
   <name>Magdalena Godlewska</name>
   <position>Doktorantka Katedry Inteligentnych Systemów
   Interaktywnych</position>
   <description>Autorka rozprawy</description>
   <contact>
      <email-address>magdal@eti.pg.gda.pl</email-address>
      <phone-number>+48583471378</phone-number>
   </contact>
   <communications>
      <simplex>Informacja o˜sprawdzeniu rozdziału przez email</simplex>
   </communications>
</worker>
```

[2]Osoby niezatrudnione w organizacji, ale uczestniczące w procesie również są ujęte w komponencie pracowników wiedzy, gdyż mogą mieć wpływ na przebieg procesu i wykonują w nim określone czynności, np. świadkowie w procesie sądowym, audytorzy, itp.

```
<worker id="PW00002" originator="no" active="no">
   <name>Bogdan Wiszniewski</name>
   <position>Kierownik Katedry Inteligentnych Systemów
   Interaktywnych</position>
   <description>Promotor rozprawy</description>
   <contact>
      <email-address>bowisz@eti.pg.gda.pl</email-address>
      <phone-number>+48583471089</phone-number>
      <ip-address>153.19.48.52</ip-address>
      <webpage-url>http://www.eti.pg.gda.pl</webpage-url>
      <postal-address>80-233 Gdańsk, ul. Narutowicza
      11/12</postal-address>
   </contact>
   <communications>
      <duplex> Kontakt telefoniczny tylko w godz. 09:00-15:00</duplex>
   </communications>
</worker>
```

Powyższy opis specyfikuje dwóch pracowników wiedzy związanych z rozprawą, z których `PW00001` jest inicjatorem i aktywnym autorem, zaś `PW00002` ma domyślnie pasywną rolę w procesie. Na etapie pisania rozprawy to są wszyscy związani z nią pracownicy wiedzy, natomiast po jej zakończeniu zostaną dołączeni kolejni, np. przewodniczący komisji przewodu doktorskiego i recenzenci.

4.3.4. Komponent dokumentów składowych

Zasadniczą częścią dokumentu MIND są dokumenty składowe, które przechowują treść i pozwalają na jej wymianę między pracownikami wiedzy. Każdy dokument składowy jest opisany osobnym elementem `<part>` umieszczonym w komponencie `<parts>`, który jest komponentem dokumentów składowych (`parts.xml`). Komponent ten odpowiada zbiorowi D w modelu organizacji opartej na wiedzy *OOW* opisanej wzorem (2.1), zaś każdy `<part>` jest elementem tego zbioru.

Każdy składnik komponentu `<parts>` identyfikowany jest za pomocą unikatowej wartości atrybutu `id`, zaś wartość atrybutu `state` specyfikuje jeden z pięciu stanów w jakim może znajdować się dany element `<part>` w całym cyklu życia:

- `new`, gdy zostanie utworzony nowy dokument składowy nie zawierający jeszcze żadnej treści,
- `modified`, gdy treść dokumentu składowego została poddana pierwszej lub kolejnej edycji,

– `verified-positive` lub `verified-negative`, gdy po edycji treść dokumentu składowego została zweryfikowana z odpowiednim skutkiem,

– `done`, gdy treść dokumentu składowego została ostatecznie zamknięta do edycji.

Każdy element `<part>` musi obligatoryjnie zawierać trzy elementy podrzędne: `<title>`, czyli tytuł składnika, `<modified>` datę ostatniej modyfikacji jego treści oraz `<content>`, a więc właściwą treść dokumentu składowego, zawartą bezpośrednio w tym elemencie (zakodowaną w formacie `base64`) lub podaną w formie odsyłacza (`href`) do konkretnego pliku w strukturze katalogów rdzenia. Atrybut `type` elementu `<content>` określa typ zawartości dokumentu składowego i może to być jeden z typów MIME lub inny typ, np. zdefiniowany dla danej organizacji szablon dokumentu.

Schemat na rysunku 4.5 wskazuje, że każdy element `<part>` może zawierać także inne elementy: bardziej szczegółowy opis (`<description>`), listę z prawami dostępu do dokumentu składowego przez poszczególnych pracowników (`<access>`), a także kolekcję adnotacji związanych z dokumentem składowym (`<annotations>`) oraz zbiór innych dokumentów (plików) załączonych do danego dokumentu (`<attachments>`).

Elementy podrzędne elementu `<access>`, poprzez wprowadzenie `id` pracownika wyspecyfikowanego w komponencie `workers.xml`, pozwalają na nadawanie praw dostępu pracownikom wiedzy do poszczególnych dokumentów składowych. Umieszczenie wartości ALL w tych elementach pozwala na nadanie danych uprawnień wszystkim pracownikom. Dostępne są następujące rodzaje uprawnień:

– `<none>` brak prawa dostępu,

– `<toc>` prawo dostępu do spisu treści,

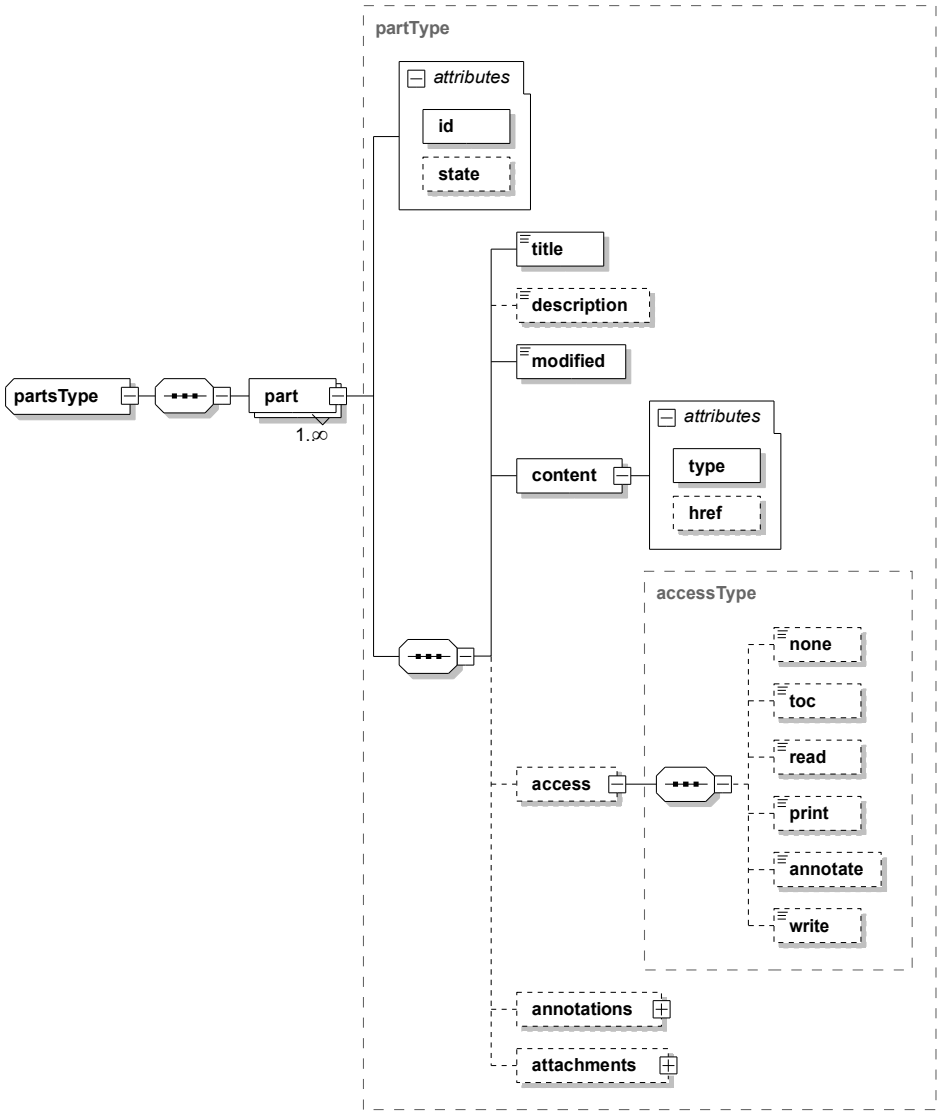
– `<read>` prawo do odczytu,

– `<print>` prawo do wydruku,

– `<annotate>` prawo nanoszenia adnotacji,

– `<write>` prawo do edycji.

Prawa te pozwalają na modyfikację ogólnych praw dostępu dla pracowników, określonych w atrybucie `active`. Każdy pracownik zdefiniowany w komponencie `workers.xml`, który otrzymał atrybut `active="yes"`, ma domyślnie przyznane prawo edycji wszystkich dokumentów składowych danego dokumentu MIND – chyba że



Rys. 4.5. Struktura logiczna komponentu dokumentów składowych

w elemencie `<part>` został zamieszczony element `<write>`, wówczas prawo do edycji mają tylko pracownicy, których `id` jest ujęte w tym elemencie. Podobnie, pracownicy z atrybutem `active="no"` mają domyślnie prawo do odczytu, które również można dowolnie modyfikować.

W tej rozprawie każdy rozdział stanowi osobny dokument składowy, zatem komponent `parts.xml` dla dwóch pierwszych rozdziałów wygląda następująco:

```
<part id="PAR00001" state="modified">
  <title>Rozdział 1: Dokument elektroniczny</title>
  <description>Rozwój dokumentów elektronicznych</description>
  <modified>2013-02-09T22:30:00</modified>
  <content type="text/x-tex"
```

```
            href="/HUB00001/Parts/dok_elektr.tex"/>
    <access>
      <annotate>PW00002</annotate>
      <print>ALL</print>
    </access>
  </part>
  <part id="PAR00002" state="new">
    <title>Rozdział 2: Organizacje oparte na wiedzy</title>
    <modified>2013-03-09T12:45:00</modified>
    <content type="text/x-tex"
            href="/HUB00001/Parts/oow.tex"/>
    <access>
      <annotate>PW00002</annotate>
      <print>ALL</print>
    </access>
  </part>
```

Z powyższego zapisu wynika, że rozdział 1. był już edytowany i ostatnie zmiany zostały wprowadzone 2013-02-09 o 22:30:00, poza tym jest to dokument typu MIME `text/x-tex` i jego zawartość znajduje się w katalogu `Parts` należącym do rdzenia `HUB00001`. Rozdział 2. został utworzony 2013-03-09 o 12:45:00 i nie był jeszcze edytowany na co wskazuje atrybut `state="new"`. Oba rozdziały mają ustalone prawa dostępu, pozwalające pracownikowi wiedzy o `id="PW00002"` na dodawanie adnotacji oraz umożliwiające wydruk dokumentów wszystkim pracownikom, na co wskazuje zapis `<print>ALL</print>`

W trakcie realizacji polityki migracji, dokumenty składowe mogą być kopiowane, scalane, dzielone na mniejsze części lub przekształcane na inne typy. Przykładowo, dokument `PAR00001` po przesłaniu jego kopii do pracownika `PW00002` (promotora) może mieć następujący opis w elemencie `<part>`:

```
  <part id="PAR00011" state="modified">
    <title>Rozdział 1: Dokument elektroniczny</title>
    <description>Rozwój dokumentów elektronicznych</description>
    <modified>2013-02-15T15:30:00</modified>
    <content type="application/pdf">xtYcVBp ... tyFFuGa1</content>
    <access>
      <annotate>PW00002</annotate>
      <print>ALL</print>
    </access>
  </part>
```

Dokument został przekształcony na format PDF, otrzymał typ opisany zgodnie z MIME jako `application/pdf`, a jego zawartość została zapisana wewnątrz elementu `<content>` i zakodowana w formacie `base64`.

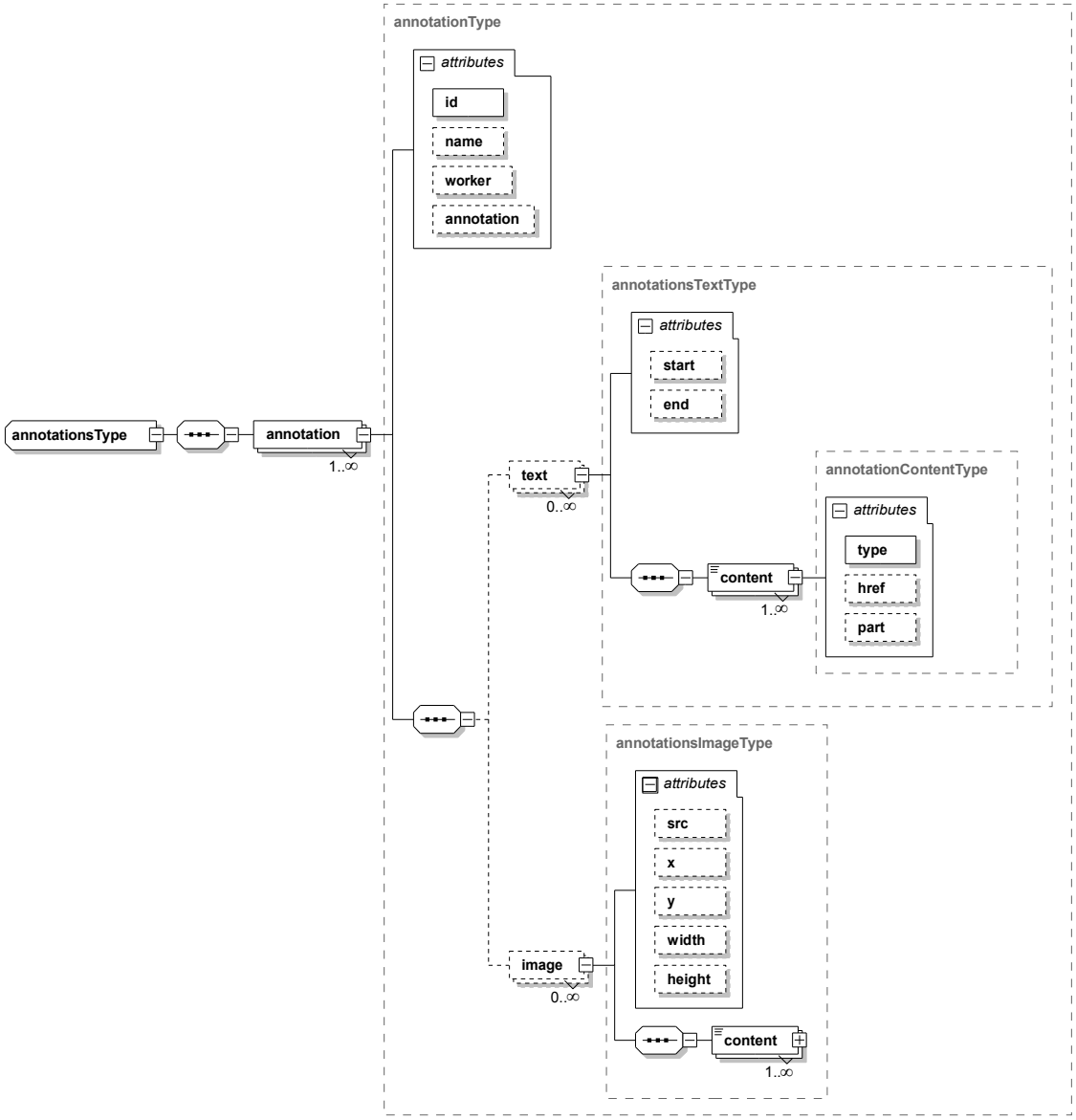
Każdy dokument składowy MIND może mieć adnotacje, dowiązywane dynamicznie w dowolnej chwili cyklu życia dokumentu. Dołączanie adnotacji jest często funkcjonalnością lokalnej aplikacji użytkownika, np. aplikacja Adobe Reader [Ado13] pozwala na dodawanie adnotacji do dokumentów PDF. Adnotacje stanowią wówczas część zawartości dokumentu ujętej w elemencie `<content>`. Architektura MIND umożliwia natomiast wprowadzane adnotacji i załączników do treści dokumentu, które również mogą być rozproszonymi dokumentami składowymi. Schemat adnotacji jest zaprezentowany na rysunku 4.6.

Atrybuty elementu `<annotation>` pozwalają na jej identyfikację poprzez unikatowy identyfikator (`id`) i nazwę (`name`) oraz wskazują na pracownika, który ją zamieścił (`worker`). Adnotacje domyślnie odnoszą się do dokumentu składowego opisanego przez dany `<part>`. Atrybut `annotation` pozwala natomiast powiązać adnotację z inną adnotacją zamieszczoną w bieżącym `<part>`.

Adnotacje mogą odnosić się do konkretnych miejsc w tekstowej treści dokumentu lub na obrazie dokumentu i reprezentują to odpowiednio elementy podrzędne `<text>` i `<image>`. Element `<content>` jest jednakowy dla obu rodzajów adnotacji i opisuje ich zawartość, która może być dowolnego typu. Najprostszym typem zawartości adnotacji jest zwykły tekst (np. typ MIME `text/plain`) wstawiony bezpośrednio do elementu `<content>`, ale adnotacją może być też dowolny plik, którego lokalizację wskazuje atrybut `href` lub inny dokument składowy określony przez atrybut `part`.

Adnotacje elementu `<part>` mogą być opisane w następujący sposób:

```
<annotations>
  <annotation id="ANN00001" worker="PW00002">
    <text start="724" end="764">
      <content type="text/plain"> Koniecznie sprawdzić, czy wszystkie
      obcojęzyczne wyrazy zostały wyróżnione kursywą!</content>
    </text>
  </annotation>
  <annotation id="ANN00002" name="nowe logo PG" worker="PW00002">
    <image src="PAR00003/1" x="0" y="0" width="10" height="40">
      <content type="image/png"
      href="http//pg.gda.pl/.../nowe-logo.png"/>
    </image>
  </annotation>
  <annotation id="ANN00003" name="komentarz" worker="PW00002">
    <text start="30" end="1034">
```

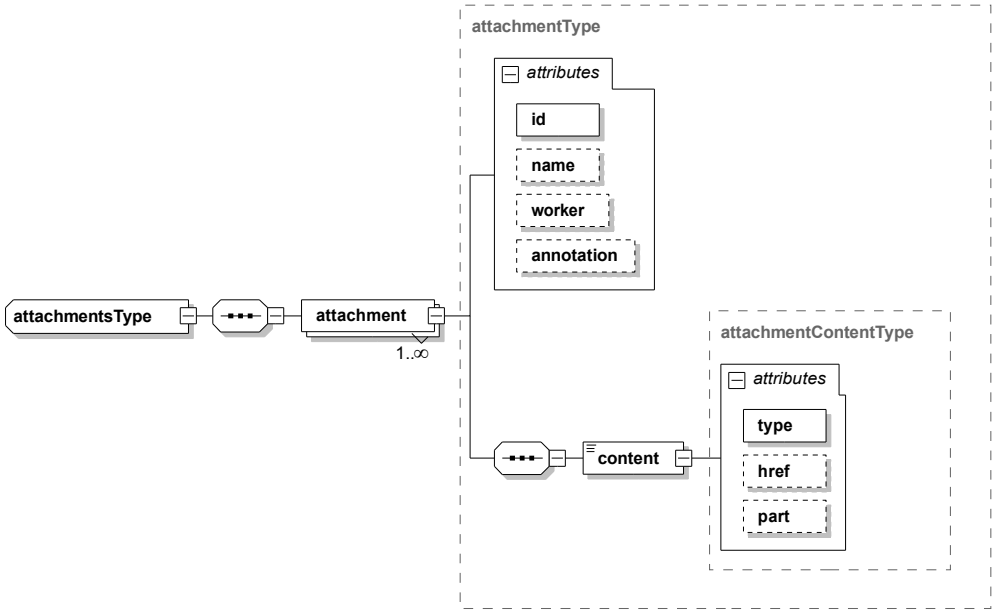


Rys. 4.6. Schemat logiczny adnotacji dołączanych do dokumentu składowego

```
        <content type="application/pdf" part="PAR00015"/>
      </text>
    </annotation>
  </annotations>
```

Struktura logiczna załączników przedstawiona na rysunku 4.7 jest zbliżona do adnotacji, z tą różnicą, że załączniki są związane z całym dokumentem składowym i nie ma konieczności wyznaczania dla nich konkretnych miejsc w dokumencie.

Zaprezentowana struktura adnotacji pozwala na dynamiczne i warstwowe dodawanie adnotacji. Jednakże możliwości komentowania treści dokumentów są ograni-



Rys. 4.7. Schemat logiczny załączników dokumentu składowego

czone do jednego obszaru w jednym dokumencie składowym. Architekturę MIND można rozszerzyć o dodawanie adnotacji powiązanych z wieloma rozproszonymi dokumentami składowymi i ustalenia dla nich dodatkowych polityk migracji. Koncepcje różnego typu adnotacji rozproszonych zaprezentowałam w publikacji [God08]. Zagadnienie to stanowi możliwe rozwinięcie omawianej otwartej architektury i nie zostało podjęte w rozprawie.

4.3.5. Komponent przepływu pracy

Komponent przepływu pracy (`path.xml`) definiuje ścieżkę migracji dokumentów składowych, która jest zapisana w standardowym formacie XPDL opartym na notacji XML. Format ten jest systematycznie rozwijany przez WfMC, międzynarodową organizację zrzeszającą głównych producentów oprogramowania do modelowania i zarządzania procesami biznesowymi. Najnowsza wersja specyfikacji XPDL 2.2 została opublikowana w roku 2012 [WfM12b] i dostosowuje język XPDL do standardu BPMN 2.0 opublikowanego w roku 2011 [Obj11]. Konsorcjum WfMC przyjęło „przyrostowy" sposób rozwoju standardu, co zapewnia trwałość rozwiązania, tzn. kompatybilność definiowanych ścieżek migracji dokumentów z przyszłymi systemami wspierającymi realizację przepływów pracy (ang. *forward compatibility*).

W języku XPDL znajdują się elementy bezpośrednio związane z realizacją procesu biznesowego oraz elementy, które pozwalają na odwzorowanie graficznych składników BPMN, np. elementy `NodeGraphicsInfo` czy `ConnectorGraphiscInfo`

są zdefiniowane w XPDL kolejno do opisu wyglądu graficznego węzłów (czynności, „torów pływackich") i obiektów łączących (tranzycji, przepływu wiadomości), co pozwala na eksport procesów tworzonych w edytorach BPMN do notacji XPDL. Do opisu ścieżki migracji dokumentów MIND wystarcza niewielki podzbiór elementów dostępnych w XPDL. W tej części rozprawy przedstawię graficzne reprezentacje struktur XPDL, które są wykorzystywane w MIND do opisu przepływu pracy.

Element główny komponentu przepływu pracy został przedstawiony na rysunku 4.8. Definiuje on elementy globalne dla całej ścieżki migracji i pozwala na dodawanie nowych poprzez element `<ExtendedAttributes>`. Element `<Performers>`, określający uczestników globalnie dla całej ścieżki migracji, jest powiązany z komponentem pracowników wiedzy (por. podrozdział 4.3.3.), zaś element `<Applications>` jest powiązany z komponentem usług (por. podrozdział 4.3.2.).

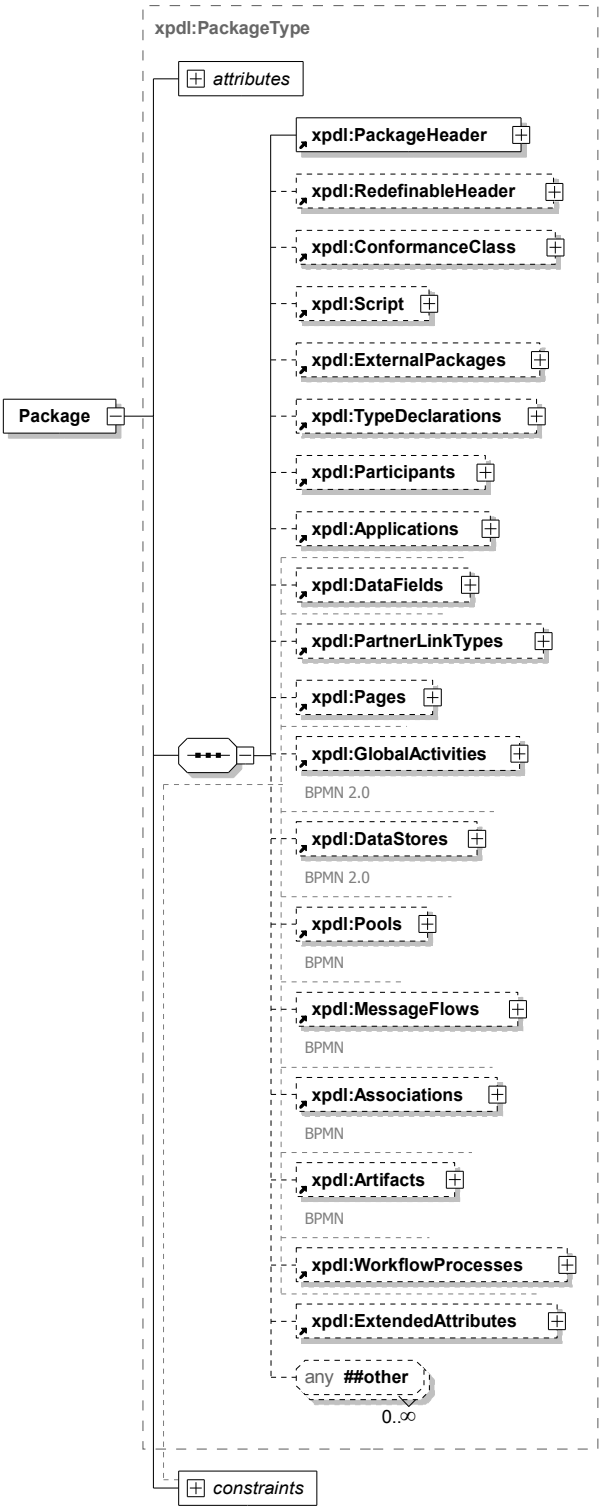
Element `<WorkflowProcesses>` zawiera definicje procesów przepływu pracy, których może być dowolnie wiele w jednym komponencie opisującym ścieżkę. W MIND jeden proces jest oznaczany jako główny, zaś pozostałe procesy mogą być wykonywane jako podprocesy procesu głównego. Proces główny jest oznaczany poprzez dodatkowy atrybut (`ExtendedAttribute`) o nazwie `MAIN_PROCESS` – to rozszerzenie jest potrzebne z powodu rozproszenia dokumentu MIND i lokalnej interpretacji ścieżki migracji na urządzeniach pracowników wiedzy, co będzie dokładnie omówione w rozdziale 5.

Elementy `<ExtendedAttribute>` można dodawać do wielu elementów XPDL, co pozwala na dostosowanie pliku XPDL do konkretnej architektury czy systemu realizującego przepływ pracy. Specyfikację tego elementu prezentuje rysunek 4.9. Poniższy zapis w XPDL wskazuje, że głównym procesem danego komponentu przepływu pracy, jest proces o `Id="WP0001"`:

```
<xpdl:ExtendedAttributes>
  <xpdl:ExtendedAttribute Name="MAIN_PROCESS" Value="WP0001"/>
</xpdl:ExtendedAttributes>
```

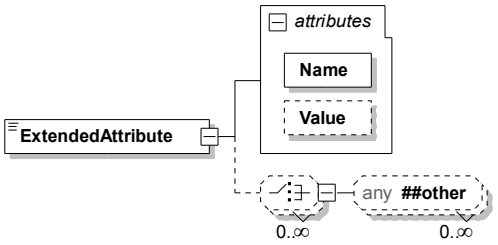
Specyfikacja elementu przepływu pracy (`<WorkflowProces>`) została przedstawiona na rysunku 4.10. Każdy element `<WorkflowProcess>` reprezentuje pewien proces, składający się z czynności `<Activities>` i przejść między nimi, czyli tranzycji `<Transitions>`. Opcjonalnie może zawierać także opis parametrów wejściowych i wyjściowych (`<FormalParameters>`), danych procesu (`<DataFields>`) czy podprocesów dostępnych tylko w danym procesie w formie zbioru czynności (`<ActivitySets>`). Dla danego procesu można określić także dodatkowych uczestników (`<Performers>`) i usługi (`<Applications>`).

Przykładowy proces edycji rozprawy został przedstawiony na rysunku 4.11



Rys. 4.8. Element główny komponentu ścieżki

w postaci diagramu BPMN. Elementy BPMN użyte na diagramie są opisane na rysunku 5.3 (str. 103). Język XPDL jest zgodny z notacją BPMN, więc istnieje



Rys. 4.9. Specyfikacja dodatkowych atrybutów komponentu ścieżki

możliwość wyeksportowania tego diagramu do pliku XPDL. Dostępne są narzędzia, które dają możliwość tworzenia diagramów BPMN i zapisywania ich w postaci XPDL, np. Together Workflow Editor [Tog11b] czy Yaoqiang XPDL Editor [Yao11].

Przykładowy proces edycji rozprawy uwzględnia następujące rodzaje czynności:

– czynność początkową i końcową, są to czynności należące do grupy `<Event>` związanej ze zdarzeniami procesu;

– czynności związane z przepływem (`<Route>`), tzw. bramki, które rozdzielają i łączą wątki przepływu pracy;

– czynności implementacyjne (`<Implementation>`) opisują określone akcje procesu związane z przetwarzaniem treści dokumentu; wykonywane są najczęściej przez pracownika wiedzy z wykorzystaniem odpowiednich narzędzi.

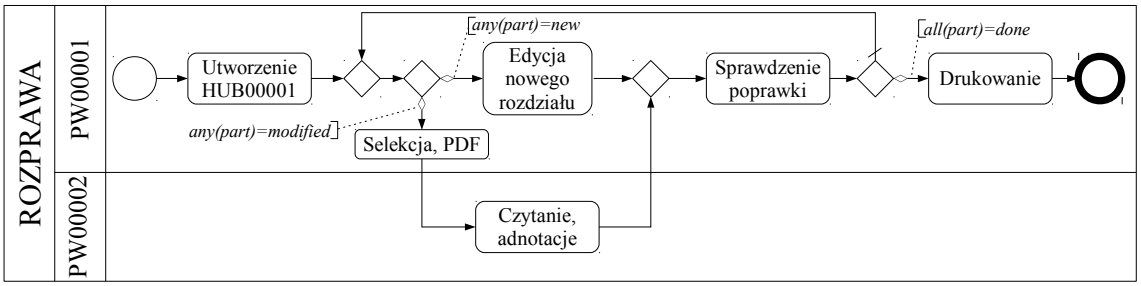
Specyfikacja elementu `<Activity>` jest przedstawiona na rysunku 4.12.

Czynności implementacyjne określają akcje, jakie mają być wykonane w danym punkcie procesu na dokumentach składowych. W szczególności mogą to być pojedyncze zadania (`<Task>`) lub podprocesy (`<Subflow>`) stanowiące referencję do kolejnego elementu `<WorkflowProcess>`. XPDL 2.2 przewiduje 9 elementów podrzędnych elementu `<Task>`. Są one przedstawione na rysunku 4.13. W MIND element `<TaskApplication>` odnosi się poprzez identyfikator do usług zdefiniowanych w komponencie usług i tam następuje ich podział na wbudowane, lokalne i zdalne oraz podane są dane potrzebne do ich uruchomienia. Element `<TaskManual>` wyznacza zaś te czynności, które są wykonywane przez pracownika bez przypisania żadnej usługi.

Na przykład, czynność Czytanie, adnotacje (rysunek 4.11) przypisana pracownikowi `PW00002` (promotorowi) ma w XPDL postać:

Rys. 4.10. Specyfikacja procesu przepływu pracy

```
<xpdl:Activity Id="ACT00007" Name="Czytanie, adnotacje">
    <xpdl:Implementation>
        <xpdl:Task>
            <xpdl:TaskApplication Id="SER00007"/>
```



Rys. 4.11. Przykładowy proces przepływu pracy dla rozprawy

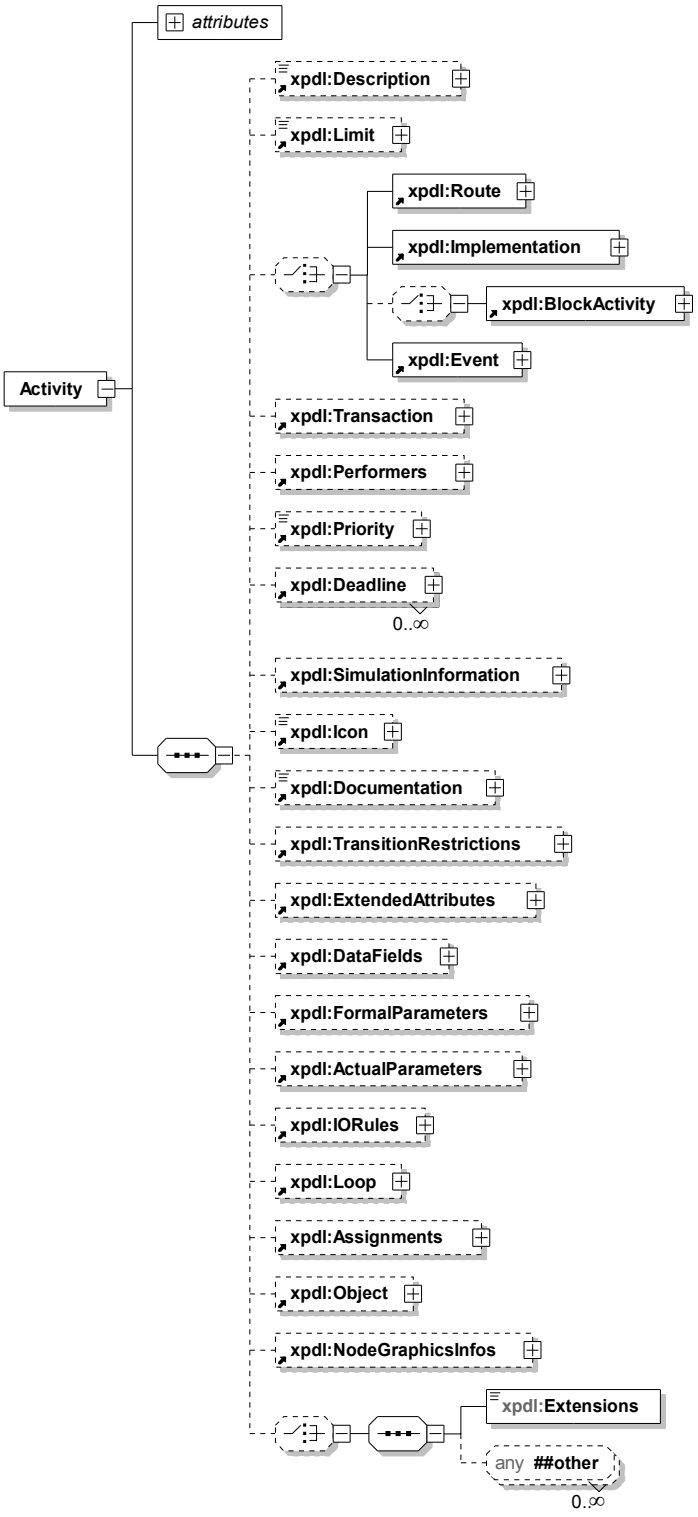
```
            </xpdl:Task>
        </xpdl:Implementation>
        <xpdl:Performers>
            <xpdl:Performer>PW00002</xpdl:Performer>
        </xpdl:Performers>
    </xpdl:Activity>
```

Zatem do wykonania tej czynności przez `PW00002` zostanie użyta aplikacja o `Id="SER00007"`, co wskazuje na konkretną usługę zdefiniowaną w komponencie usług dokumentu (zob. str. 74). Jest to usługa lokalna służąca do odczytu dokumentów PDF i umożliwiająca nanoszenie adnotacji.

Po wykonaniu każdej czynności przedstawionej na diagramie 4.11 następuje jeszcze jedna czynność, która zmienia atrybut `state` dokumentów składowych (rys. 4.5 str. 79), czynność ta została pominięta dla uproszczenia diagramu. Po utworzeniu rdzenia dokumentu `HUB00001`, każdy dokument składowy (rozdział rozprawy) otrzymuje atrybut `state="new"`. Edycja nowego rozdziału zmienia jego stan na `modified`. Promotor po przeczytaniu i umieszczeniu komentarzy do danego rozdziału, może ustalić jego stan na `verified-negative` (wówczas wymaga kolejnego czytania przez promotora) lub `verified-positive` (wówczas nawet jeśli są jakieś adnotacje, nie wymaga już kolejnego czytania). Czynność „Sprawdzenie, poprawki" może zmienić stan rozdziału na `done`, tylko jeśli otrzymał wcześniej stan `verified-positive`, w przeciwnym przypadku ustala stan na `modified`.

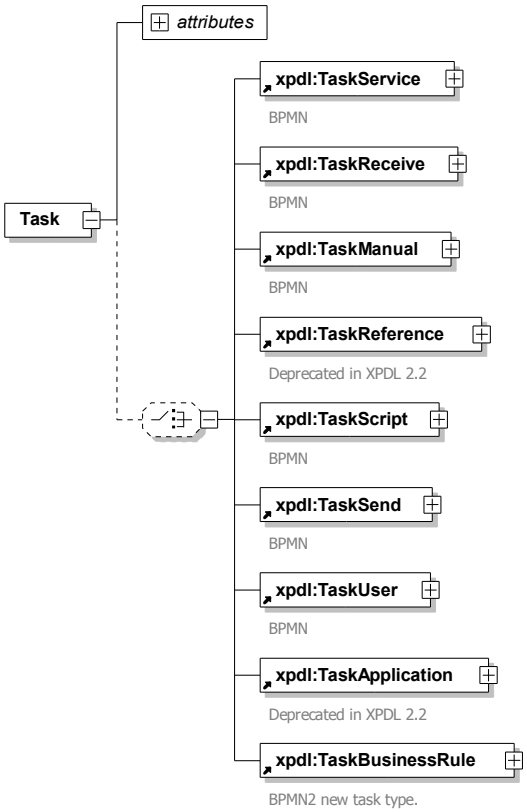
Czynności związane z przepływem (element `<Route>`), tzw. bramki (ang. *gateway*) nie są związane z żadnymi zadaniami wykonywanymi na dokumencie. Służą one do rozdzielania (element `<split>`) lub łączenia (element `<join>`) wątku przepływu pracy. XPDL przewiduje możliwość zastosowania 4 typów bramek: `Exclusive`, `Inclusive`, `Parallel` i `Complex`. W elemencie podrzędnym `<TransitionRestriction>` przedstawionym na rysunku 4.14 określa także, jakie tranzycje są związane z daną bramką.

Na diagramie 4.11. druga bramka jest typu `Parallel`, co pozwala na zrównole-



Rys. 4.12. Specyfikacja czynności procesu

glenie pracy uczestników procesu. Dzięki temu, w tym samym czasie promotor może czytać już napisane rozdziały, zaś doktorantka może pracować nad treścią nowego

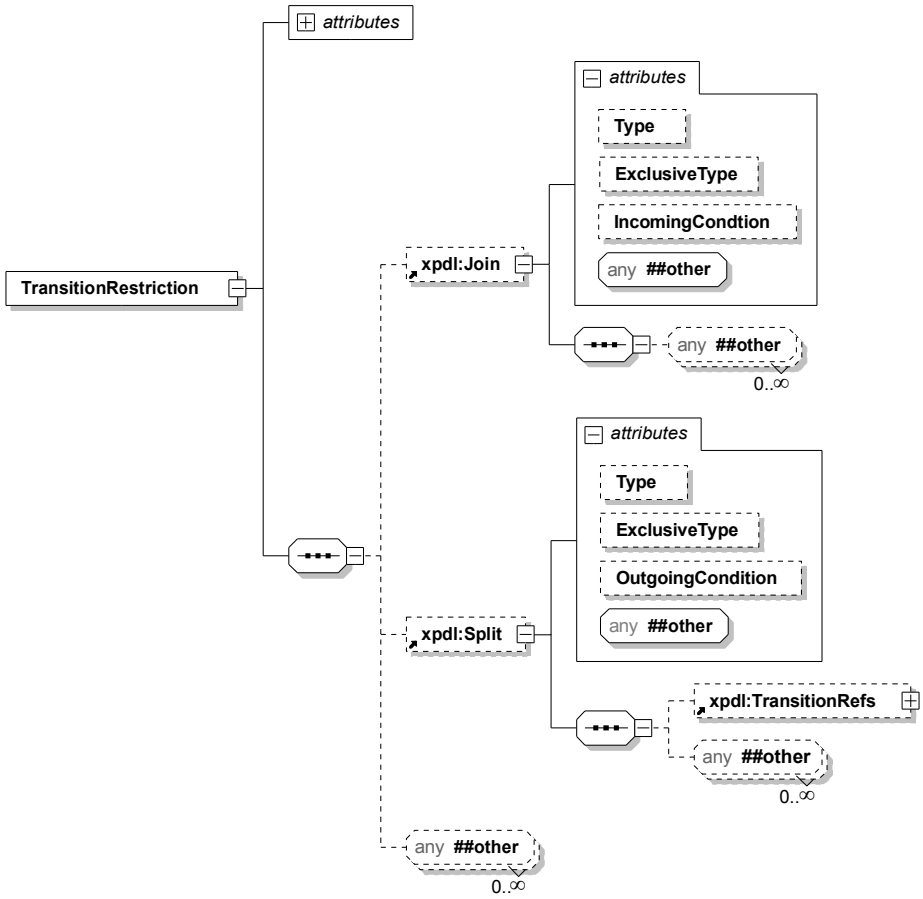


Rys. 4.13. Specyfikacja rodzaju zadań wykonywanych w ramach czynności

rozdziału. Zapis tej bramki w XPDL wygląda następująco:

```
<xpdl:Activity Id="ACT00004">
    <xpdl:Route GatewayType="Parallel"/>
    <xpdl:TransitionRestrictions>
        <xpdl:TransitionRestriction>
            <xpdl:Split Type="Parallel">
                <xpdl:TransitionRefs>
                    <xpdl:TransitionRef Id="TRS00004"/>
                    <xpdl:TransitionRef Id="TRS00005"/>
                </xpdl:TransitionRefs>
            </xpdl:Split>
        </xpdl:TransitionRestriction>
    </xpdl:TransitionRestrictions>
</xpdl:Activity>
```

Bramka `Split` typu `Parallel`, jeśli jest związana z tranzycjami bezwarunkowymi, rozdziela wątek przepływu pracy i przekierowuje go na wszystkie tranzycje wychodzące, wymienione w elemencie `<TransitionRestriction>`. Warunkowe tranzycje pozwalają natomiast na dodatkową selekcję ścieżki przepływu pra-



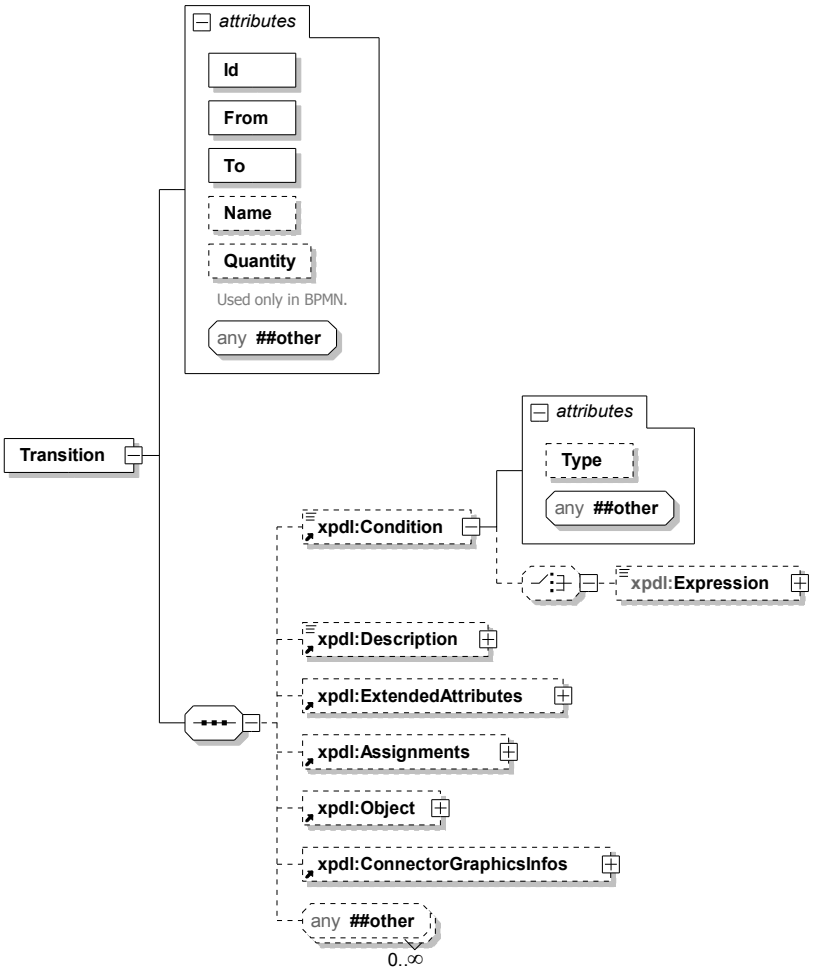
Rys. 4.14. Specyfikacja rodzajów synchronizacji

cy. Element `<Condition>`, dodawany w tranzycjach warunkowych, może zawierać atrybut `type` przyjmujący jedną z wartości: `condition`, `otherwise`, `exception` i `defaultexception`, który określa rodzaj warunku danego przejścia, przy czym `condition` jest wartością domyślną. Rysunek 4.15 prezentuje specyfikację elementu opisującego tranzycje.

Przedstawiony przykład bramki typu `Parallel` rozdziela wątek na dwie tranzycje: `TRS00004` i `TRS00005`. Obie te tranzycje są warunkowe (wyjaśnienie użytych symboli BPMN znajduje się rys. 5.3 str. 103). Zapisane w XPDL wyglądają następująco:

```
<xpdl:Transition From="ACT00004" Id="TRS00004" To="ACT00006">
    <xpdl:Condition>any(part) = new</xpdl:Condition>
</xpdl:Transition>

<xpdl:Transition From="ACT00004" Id="TRS00005" To="ACT00005">
    <xpdl:Condition>any(part) = modified</xpdl:Condition>
</xpdl:Transition>
```



Rys. 4.15. Specyfikacja tranzycji w przepływie pracy

Dzięki tym warunkom, czynność Edycja nowego rozdziału (`ACT00006`) zostanie uaktywniona tylko wtedy, kiedy będzie istniał dokument składowy o stanie `new`, zaś promotor otrzyma do przeczytania pracę, jeśli któryś dokument składowy ma stan `modified`. Czynność Selekcja, PDF realizuje w zasadzie dwie czynności, połączone dla uproszczenia diagramu. Pierwsza jest powiązana z usługą wbudowaną, jaką jest transformacja rozdzielająca rozdziały i pozwalająca na wybór tylko tych rozdziałów, które mają stan `modified`, tak by promotor nie otrzymywał nowych ani już zatwierdzonych rozdziałów. Druga dokonuje konwersji dokumentu w formacie LaTeX do formatu PDF.

Przedstawiona architektura MIND zapewnia rozproszenie, mobilność oraz interaktywność dokumentu. Rozproszenie wynika z podziału dokumentu na części składowe, które dostarczane są w sposób niezależny odpowiednim pracownikom na ich lokalne urządzenia w formie odrębnych pakietów MIND, które zawierają

w sobie wszelkie niezbędne informacje do ich lokalnego przetwarzania. Mobilność wynika z wbudowanej w każdy dynamiczny komponent dokumentu polityki migracji, na podstawie której potrafi wyznaczyć i zrealizować swój przepływ pracy, przenosząc w ten sposób informacje między pracownikami wiedzy. Interaktywność zaś związana jest z komponentem usług, który dostarcza odpowiednią funkcjonalność, dostosowującą dokument do pracy w lokalnym środowisku i wspierającą pracownika w edycji dostarczonej treści.

Architektura MIND została opracowana z myślą o zapewnieniu komunikacji treści oraz koordynacji czynności dokumentu. Komunikacja treści jest zrealizowana poprzez umożliwienie dodawania do dokumentu MIND zawartości dowolnego typu oraz wspieranie jej odtwarzania i edycji przez różnych pracowników poprzez dołączone usługi. Dzięki temu, że dokument wskazuje usługę przeznaczoną do przetwarzania jego treści, pozwala to na eliminację dwóch problemów odpowiedniego przetwarzania zawartości, przedstawionych w podrozdziale 1.1.2.

Koordynacja czynności jest zrealizowana poprzez wbudowaną w dokument ścieżkę migracji opisującą przepływ dokumentu i wyznaczającą konkretne zadania pracownikom. Szczegółowy opis zagadnienia zarządzania przepływem pracy w oparciu o architekturę MIND przedstawiam w kolejnym rozdziale rozprawy.

Rozdział 5

Koordynacja czynności rozproszonego dokumentu elektronicznego

Jedną z cech opisanej w poprzednim rozdziale architektury MIND jest fizyczne rozproszenie komponentów odpowiedzialnych za realizację czynności procesu biznesowego. Proces ten jest wdrażany dynamicznie na niezależnych urządzeniach osobistych pracowników wiedzy. Dokument MIND ma wbudowaną zarówno ścieżkę migracji definiującą proces (komponent przepływu pracy), jak i funkcjonalność dostarczaną przez usługi (komponent usług). Komponenty te, w cyklu życia dokumentu, stają się agentami, które są autonomiczne, a więc niezależne od określonych platform wspierających zarządzanie przepływem pracy oraz mobilne, a więc zdolne do uruchamiania poszczególnych czynności na osobistych urządzeniach pracowników w zależności od potrzeby wynikającej z przebiegu procesu.

Przenoszenie w strukturze dokumentu opisu ścieżki migracji oraz usług potrzebnych do jej realizacji pozwala na rozproszone zarządzanie przepływem pracy, wykonywane niezależnie na osobistych urządzeniach pracowników. Wymaga to zainstalowania lokalnej usługi zarządzającej przepływem pracy – klienta LWE (ang. *local workflow engine*), która w większości przypadków wymaga łączności w Internetem jedynie w celu odbierania i wysyłania dokumentów. Każdy klient LWE jest zatem składnikiem luźno powiązanego (ang. *loosely coupled*) systemu rozproszonego skoncentrowanego na dokumencie MIND, służącego do komunikacji jego treści i koordynacji wykonywanych na nim czynności.

Ten rozdział rozprawy opisuje ideę rozproszonego zarządzania przepływem pracy w oparciu o dokument MIND i usługi specjalnie zaprojektowanego klienta pocztowego LWE. Odnosi się także do wzorców koordynujących czynności w systemach

poczty elektronicznej opisanych w podrozdziale 3.2, przedstawiając sposób ich realizacji w ramach architektury MIND.

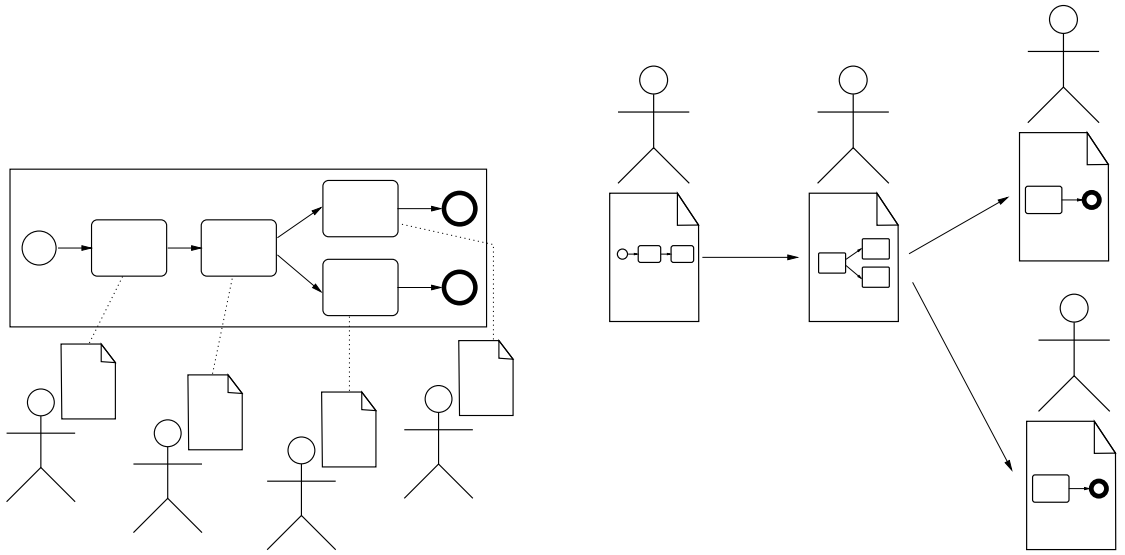
5.1. Rozproszone zarządzanie przepływem pracy

W modelu referencyjnym przepływu pracy [WfM99] *usługa zarządzająca* przepływem pracy (ang. *workflow enactment services*) ma za zadanie interpretację i wykonanie procesu oraz koordynowanie jego czynności. Może się składać z jednego lub kilku współpracujących ze sobą „silników" (ang. *workflow engines*), czyli usług/aplikacji udostępniających środowisko wykonawcze dla procesów przepływu pracy. Jeśli w celu realizacji procesu współpracuje ze sobą kilka „silników" przepływu pracy, to dane sterujące procesem muszą być dla nich wszystkich dostępne. Dlatego najczęściej implementuje się scentralizowane usługi zarządzające przepływem pracy. Nie wyklucza to, natomiast, rozproszenia uczestników procesu, którzy mogą wymieniać dane potrzebne aplikacji realizującej przepływ pracy, przesyłając wiadomości poprzez sieć.

W podrozdziale 1.2.3 została opisana koncepcja warstwy pośredniczącej (ang. *middleware*) pomiędzy rozproszonymi dokumentami-agentami a różnymi aplikacjami dostępnymi na rozproszonych lokalizacjach. Przykładem wykorzystania warstwy pośredniczącej jest m.in. koordynacja czynności wykonywanych na dokumentach w środowisku rozproszonym. Koncepcja ta została zrealizowana w modelach XMIDDLE [MCZE02] i LIME [MPR06] opartych o model Linda – współdzielonej przestrzeni danych (przestrzeni krotek). Model przestrzeni krotek został następnie wykorzystany do synchronizacji przepływu sterowania w rozproszonym systemie realizującym przepływ pracy [MWL08]. Mimo że wspomniane rozwiązania pozwalają na współpracę rozproszonych, luźno powiązanych „silników", to koordynacja wykonywanych czynności wymaga mechanizmów komunikacji komponentów procesu z warstwą pośredniczącą, która stanowi, de facto, centralną jednostkę zarządzającą procesem. Rozwiązania te są zatem zależne od połączenia internetowego oraz wymagają implementacji specjalnych protokołów zapewniających spójność przestrzeni krotek w przypadku utraty tego połączenia.

Rysunek 5.1 prezentuje różnice między przedstawionymi usługami zarządzającymi przepływem pracy, a usługą proponowaną do zarządzania polityką migracji dokumentów MIND. Nazwałam je odpowiednio: *usługą centralną* i *usługą rozproszoną*.

Rozproszona usługa zarządzająca przepływem pracy, zalecana do realizacji polityki migracji dokumentów MIND, posiada zalety modelu skoncentrowanego na dokumencie (zob. podrozdziały 1.2 i 2.2) oraz zakłada umieszczenie w strukturze jednego



(a) Usługa centralna (b) Usługa rozproszona

Rys. 5.1. Usługi zarządzające przepływem pracy dokumentu

dokumentu wszystkich trzech typów danych określonych w modelu referencyjnym przepływu pracy [WfM99]:

- sterujących przepływem, np. stan procesu, aktualnie wykonywana czynność, itp.
- określających strukturę przepływu, np. tranzycje, warunki przejść, przypisanie uczestników do czynności, itp.
- pośredniczących między zewnętrznymi aplikacjami, np. parametry wejściowe i wyjściowe aplikacji, itp.

Dzięki temu zarówno komunikacja treści, jak i koordynacja czynności procesu, może odbywać się za pośrednictwem poczty elektronicznej, jako uniwersalnej, powszechnej i sprawdzonej warstwy pośredniczącej. Wymaga to implementacji niewielkiego rozmiaru aplikacji, która zainstalowana na lokalnym urządzeniu każdego uczestnika procesu, pełni rolę „lekkiego" klienta pocztowego oraz usługi zarządzającej przepływem pracy dokumentu MIND. W dalszej części rozprawy tę aplikację będę określać skrótem LWE (ang. *local workflow engine*).

Poza tym, na podstawie analizy wzorców przepływu sterowania przeprowadzonej w podrozdziale 3.2.2, można wykazać, że wymiana dokumentów pomiędzy poszczególnymi klientami LWE zainstalowanymi na urządzeniach pracowników wiedzy, pozwala na skuteczne zarządzanie procesami realizowanymi przez organizacje oparte

na wiedzy. Zaledwie kilka z przeanalizowanych wzorców wymaga przesyłania sygnałów dodatkowymi kanałami komunikacyjnymi, nie mniej jednak również można je zaimplementować w oparciu o standardowe protokoły pocztowe. Sposób realizacji poszczególnych wzorców jest dokładnie opisany w podrozdziale 5.3.

Poprawna implementacja komponentu przepływu pracy (zob. podrozdział 4.3.5) czyni dokumenty MIND mobilnymi agentami i pozwala na realizację ich polityki migracji. Komponent ten powinien spełniać dwa istotne wymogi:

1. dawać możliwość realizacji kanonicznego zbioru wzorców koordynujących czynności w systemach poczty elektronicznej wymienionego w tabeli 3.2 (str. 63),

2. być na tyle małym plikiem danych, aby można go było bez trudu przesyłać pocztą elektroniczną i przetwarzać na urządzeniach przenośnych o małej mocy obliczeniowej.

Komponent przepływu pracy w obecnej implementacji MIND jest dokumentem XPDL. Format ten pozwala na definicję wszystkich trzech wymienionych powyżej typów danych określonych w modelu referencyjnym przepływu pracy. W szczególności, pozwala na dodawanie dowolnych zmiennych poprzez elementy `<ExtendedAttribute>`, dzięki czemu można przechowywać w dokumencie dane sterujące przepływem, potrzebne do prawidłowej interpretacji komponentu przepływu pracy przez klienta LWE. Jest to niezbędne, gdyż wszystkie klienty LWE uczestniczące w procesie tworzą rozproszoną usługę zarządzającą przepływem pracy przedstawioną na rysunku 5.1(b), w której stan procesu nie istnieje centralnie. Zatem po przybyciu dokumentu MIND na urządzenie pracownika, klient LWE musi zidentyfikować, którą czynność procesu ma wykonać.

Wyznaczenie odpowiedniej czynności odbywa się na podstawie dwóch dodatkowych atrybutów dodanych do pliku `path.xml`. Po pierwsze, element główny komponentu przepływu pracy (rys. 4.8) zawiera informację o tym, który element `<WorkfowProcess>` jest procesem nadrzędnym, określoną w atrybucie o nazwie `MAIN_PROCESS`. Ścieżka migracji może bowiem składać się z procesu nadrzędnego oraz wielu zagnieżdżonych podprocesów. W XPDL wszystkie elementy `<WorkfowProcess>` są umieszczone na jednym poziomie zagnieżdżenia, jako podelementy elementu `<WorkfowProcesses>`. Dopiero element czynności `<Activity>` może zawierać w sobie odwołanie do podprocesu i w ten sposób ustalana jest hierarchia procesów. Zatem do każdej czynności i podprocesu można się dostać z procesu nadrzędnego, którego identyfikator jest zapisany w atrybucie `MAIN_PROCESS`. Po drugie, każdy element `<WorkfowProcess>` zawiera informację na temat czynności, która powinna być wykonana przez klienta LWE. Informacja ta jest zawarta w dodatkowym

atrybucie procesu o nazwie `CURRENT_ACTIVITY`. W momencie definiowania rdzenia dokumentu, atrybut `CURRENT_ACTIVITY` wszystkich procesów wskazuje na czynność początkową.

Mechanizm dodatkowych atrybutów ilustruje poniższy fragment kodu XPDL dla przykładowego procesu edycji tej rozprawy (zob. rys. 4.11):

```
<xpdl:Package Id="P00001" Name="Komponent przepływu pracy dla rozprawy">
...
  <xpdl:WorkflowProcesses>
          ...
    <xpdl:WorkflowProcess Id="WP0001" Name="Rozprawa">
      ...
      <xpdl:Activities>
        ...
        <xpdl:Activity Id="ACT00006" Name="Selekcja, PDF">
          ...
          <xpdl:Performers>
            <xpdl:Performer>PW00001</xpdl:Performer>
          </xpdl:Performers>
          ...
        </xpdl:Activity>
        <xpdl:Activity Id="ACT00007" Name="Czytanie, adnotacje">
          ...
          <xpdl:Performers>
            <xpdl:Performer>PW00002</xpdl:Performer>
          </xpdl:Performers>
          ...
        </xpdl:Activity>
        ...
      </xpdl:Activities>
      <xpdl:Transitions>
        ...
        <xpdl:Transition From="ACT00006" Id="TRS00005" To="ACT00007">
             ...
      </xpdl:Transitions>
      <xpdl:ExtendedAttributes>
        ...
        <xpdl:ExtendedAttribute Value="ACT00006"
        Name="CURRENT_ACTIVITY"/>
        ...
      </xpdl:ExtendedAttributes>
```

```
    </xpdl:WorkflowProcess>
    ...
  </xpdl:WorkflowProcesses>
  <xpdl:ExtendedAttributes>
              ...
    <xpdl:ExtendedAttribute Name="MAIN_PROCESS" Value="WP0001"/>
              ...
  </xpdl:ExtendedAttributes>
</xpdl:Package>
```

W powyższym przykładzie, atrybut `MAIN_PROCESS` wskazuje, że proces o `Id="WP0001"` jest procesem nadrzędnym. W tym procesie, atrybut `CURRENT_ACTIVITY` wskazuje czynność o `Id="ACT00006"`, która jest przypisana pracownikowi wiedzy `PW00001`. Zatem klient LWE wykona czynność `ACT00006` z procesu `WP0001` na urządzeniu pracownika `PW00001`, a następnie na podstawie tranzycji `TRS00005` wyznaczy kolejną czynność i zamieni wartość `CURRENT_ACTIVITY` na `ACT00007`.

Niektóre wzorce przepływu sterowania wymagają jeszcze innych dodatkowych atrybutów, nie zaprezentowanych w powyższym przykładzie. Jednym z nich jest atrybut przechowujący licznik, którego wartość jest ustalana w momencie rozbicia ścieżki przepływu i jest następnie wykorzystywana przez klienta LWE w puncie jej łączenia, do ustalenia liczby oczekiwanych dokumentów. Atrybut blokada (ang. *lock*), to natomiast binarny semafor, który zapisuje wartość sygnału przychodzącego do klienta LWE z zewnątrz – jego wartość początkowa wynosi *true*, co blokuje wykonanie czynności przez klienta LWE i wprowadza go w stan oczekiwania na sygnał.

Umieszczenie wszystkich niezbędnych danych sterujących procesem w komponencie przepływu pracy sprawia, że dokument jest jedynym źródłem informacji o wykonywanych czynnościach przed jego przybyciem na wyznaczoną lokalizację. Zatem poszczególne klienty LWE, składające się na rozproszony system zarządzający przepływem pracy (zob rys. 5.1(b)), nie posiadają żadnej informacji o czynnościach procesu przed przybyciem dokument MIND.

Zadaniem klienta LWE jest nie tylko odczytanie danych sterujących i zmienianie wartości odpowiednich atrybutów, lecz również pośredniczenie pomiędzy dokumentem a pracownikiem wiedzy w wykonywaniu zadań przypisanych odpowiednim czynnościom. Dokument przynosi ze sobą informacje na temat operacji, jakie mają być na nim wykonane. Są one zapisane w komponencie usług i połączone z konkretnymi czynnościami. Klient LWE powinien mieć zdolność wykonywania odpowiednich skryptów przyniesionych przez dokument lub uruchamiania usług lokalnych czy zdalnych. Dzięki temu klient LWE służy do koordynacji czynności procesowych poprzez

interpretację ścieżki migracji oraz ułatwia komunikację treści poprzez uruchamianie odpowiednich usług do jej przetwarzania.

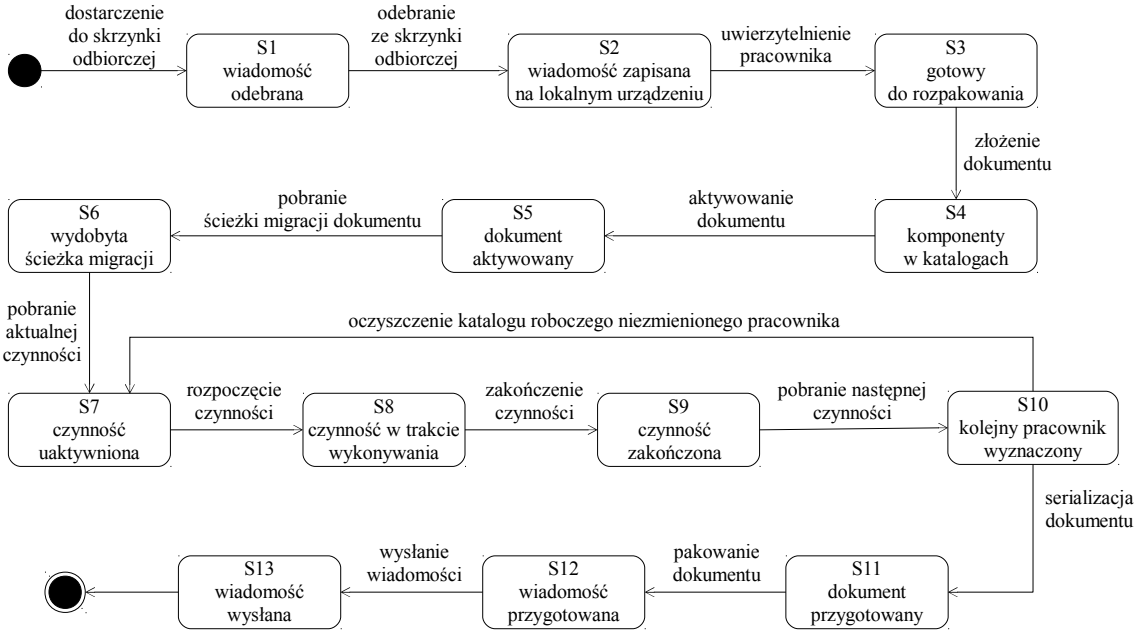
Lokalne i etapowe uruchamianie poszczególnych fragmentów komponentu przepływu pracy przez klienta LWE pozwala także na wprowadzenie edycji tego komponentu już w czasie realizacji procesu. Edycja jest wykonywana automatycznie przez klienta LWE przy zmianie wartości dodatkowych atrybutów, ale może być również umożliwiona niektórym pracownikom wiedzy do modyfikacji ścieżki migracji.

Funkcje klienta LWE mogą być również udostępnione jako usługa sieciowa, zwłaszcza użytkownikom, którzy pełnią podrzędną rolę w procesie i nie są pracownikami danej organizacji lub w sytuacji, gdy instalacja na urządzeniu osobistym jest z jakiegoś powodu niemożliwa lub niepożądana.

5.2. Podstawowa funkcjonalność klienta LWE

Dokument MIND rozpoczyna swój lokalny cykl życia na urządzeniu osobistym pracownika wiedzy z chwilą dotarcia pakietu MIND na skrzynkę odbiorczą serwera pocztowego danego pracownika i kończy go w chwili wysłania pakietu MIND kolejnemu pracownikowi. Fazy cyklu życia dokumentu na pojedynczym urządzeniu pracownika prezentuje diagram stanów zamieszczony na rysunku 5.2. Przejścia między stanami określają podstawową funkcjonalność klienta LWE.

Stan początkowy (S1) odpowiada fazie cyklu życia dokumentu, w której wiadomość zawierająca pakiet MIND zostaje dostarczona poprzez protokoły pocztowe do skrzynki odbiorczej określonego pracownika wiedzy znajdującej się na serwerze pocztowym. Niektóre wzorce przepływu pracy wymagają dostarczenia więcej niż jednego pakietu MIND. Klient LWE identyfikuje wiadomość zawierającą pakiet MIND, odbiera ją ze skrzynki odbiorczej i zapisuje na lokalnym urządzaniu pracownika, przechodząc do stanu (S2). Zanim pakiet MIND będzie gotowy do rozpakowania (S3), może być wymagane dodatkowe uwierzytelnienie pracownika. Serwery pocztowe na ogół wymagają podania nazwy konta i hasła, które uwierzytelniają odbiorcę i pozwalają mu na odbiór wiadomości, co już w przejściu do stanu (S2) pozwala sprawdzić, czy dokument trafił do właściwego odbiorcy. Mogą zostać w tym celu wykorzystane także biometryczne mechanizmy zabezpieczenia dokumentów [SSW13]. Komponenty odebranego i zapisanego pakietu MIND są umieszczane w odpowiedniej strukturze lokalnych katalogów (S4). Opis zawartości katalogu nadrzędnego znajduje się w podrozdziale 4.3. Taka postać dokumentu pozwala na jego aktywowanie przez klienta LWE poprzez wiązanie danych logicznych z obiektami. Aktywowany dokument (S5), dzięki wbudowanej funkcjonalności, może wchodzić



Rys. 5.2. Fazy cyklu życia dokumentu na pojedynczym urządzeniu pracownika wiedzy

w interakcję z klientem LWE, pracownikiem wiedzy, jego lokalnym systemem oraz usługami dostarczanymi przez zewnętrzne serwery.

Interakcja klienta LWE z dokumentem rozpoczyna się od wydobycia komponentu przepływu pracy (S6) i interpretację jego danych sterujących w celu wyznaczenia i uaktywnienia właściwej czynności, dla danego punktu procesu (S7). Uaktywniona (ang. *enabled*) czynność zostaje następnie uruchomiona i wykonana z wykorzystaniem przypisanej do niej usługi (S8). W zależności od swojego typu może być wykonana automatycznie lub poprzez interakcję z przypisanym jej pracownikiem. Czynność zostaje zakończona (S9) po wykonaniu wymaganej przez nią pracy. Czynność też może zostać zatrzymana przed jej zakończeniem, np. w sytuacji czynności łączących, które do zakończenia swojej pracy wymagają dostarczenia więcej niż jednego pakietu MIND.

Po zakończeniu czynności, wychodzące z niej tranzycje są interpretowane w celu wyznaczenia kolejnych czynności procesu. Jeśli wyznaczona jest więcej niż jedna czynność, to komponent przepływu pracy jest dla każdej z nich kopiowany i do poszczególnych kopii jest wpisywany identyfikator danej czynności jako nowa wartość atrybutu `CURRENT_ACTIVITY`. Z każdej z tych czynności odczytywani są także przypisani im pracownicy wiedzy. Zatem w tej fazie cyklu życia dokumentu wyznaczany jest zbiór kolejnych pracowników wiedzy (S10). Od tej chwili cykl życia każdego dokumentu MIND może przebiegać niezależnie.

Częstą sytuacją jest wykonywanie pewnej sekwencji czynności procesowych przez tego samego pracownika. Zatem, jeśli w kolejnej czynności określony jest ten sam pracownik, co w poprzedniej (lub w ogólnie nie jest określony np. przy czynnościach „bramkach"), wówczas jego lokalny katalog przechowujący komponenty MIND jest przebudowywany (czyszczony) w celu przygotowania komponentów dokumentu do wykonania następnej czynności (powtórzenie stanów S7,..., S10). Jeśli jednak wykonawcą kolejnej czynności jest inny pracownik, komponenty dokumentu są serializowane (S11) i tworzony jest pakiet MIND gotowy do przesłania na nową lokalizację (S12). Fazą kończącą cykl życia dokumentu jest wysłanie pakietu MIND na adres wskazanego w komponencie przepływu pracy pracownika wiedzy.

5.3. Wzorce koordynacji dokumentu

Na diagramie 5.2 została przedstawiona podstawowa funkcjonalność klienta LWE, natomiast sposób realizacji konkretnych przejść między stanami, zwłaszcza od punktu uaktywnienia czynności do wyboru kolejnego pracownika (S7,..., S10), jest zależny od konkretnych sytuacji, które wystąpiły w procesie. Sytuacje te zostały przedstawione w postaci kanonicznego zbioru wzorców koordynujących czynności w systemach poczty elektronicznej w podrozdziale 3.2.2. W tej części rozprawy opiszę funkcjonalność klienta LWE w kontekście implementowalności przedstawionych wzorców (por. tabela 3.2).

Jak wspomniałam wcześniej, sieć klientów LWE tworzy rozproszony, luźno powiązany system zarządzający przepływem pracy, który nie posiada żadnego centralnego repozytorium przechowującego globalny stan procesu przepływu pracy. W modelu organizacji opartej na wiedzy opisanym w podrozdziale 2.4, stan lokalny procesu jest zdefiniowany jako wartość funkcji γ występującej we wzorze (2.9), która przyporządkowuje aktualną czynność dokumentowi składowemu MIND. Ponieważ czynności uaktywniane są przez klienty LWE, zatem to właśnie w klientach LWE dostępna jest informacja o stanie lokalnym dokumentu składowego w procesie. Realizacja kanonicznego zbioru wzorców koordynujących czynności, którego elementy są wymienione w tabeli 3.2 (str. 63), wymaga różnej taktyki sprawdzania lokalnego stanu dokumentu i dlatego zbiór ten został podzielony na trzy kategorie:

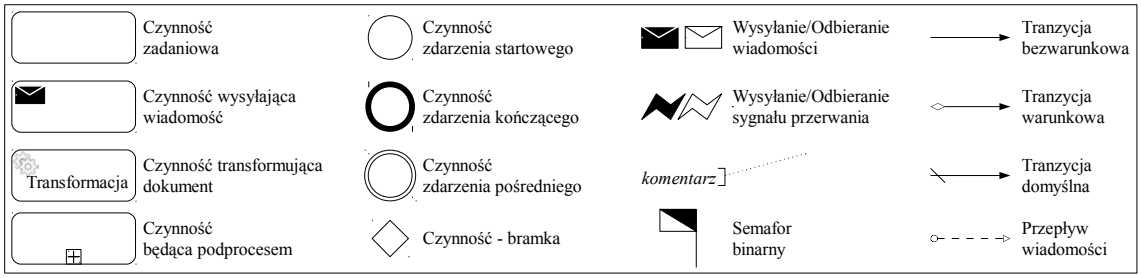
1. wzorce z *rozproszonym* stanem dokumentu, których realizacja zależy wyłącznie od lokalnego stanu dokumentu dostępnego w pojedynczym kliencie LWE,

2. wzorce ze *sprzężonym* stanem dokumentu, których realizacja zależy od dwóch lub większej liczby stanów lokalnych dokumentów dostępnych w różnych

klientach LWE,

3. wzorce z *zagnieżdżonym* stanem dokumentu, których realizacja jest związana ze stanami lokalnymi pewnego zbioru czynności zaimplementowanego jako podproces.

Implementację wcześniej opisanego kanonicznego zbioru wzorców przedstawię dalej w formie diagramów BPMN (rysunki 5.4 – 5.13), rozdzielając jego elementy względem powyższych kategorii. Zaimplementowane wzorce nazwałam *wzorcami koordynacji dokumentu*, gdyż odnoszą się do sytuacji związanych z przepływem dokumentu MIND. Dla ułatwienia interpretacji diagramów, na rysunku 5.3 zebrałam wszystkie wykorzystane symbole, stanowiące podzbiór standardowych symboli BPMN z dodatkowym symbolem, semaforem binarnym, oznaczającym stan oczekiwania przez klienta LWE na wiadomość (sygnał) z innej części procesu – mechanizm ten wykorzystuję do realizacji wzorców ze sprzężonym stanem dokumentu.



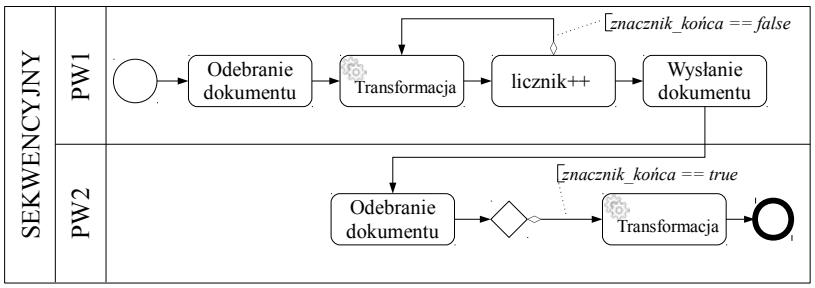
Rys. 5.3. Elementy BPMN użyte do specyfikacji wzorców koordynacji dokumentu

5.3.1. Wzorce z rozproszonym stanem dokumentu

Wzorce z rozproszonym stanem dokumentu opisują sytuacje, w których przebieg aktualnej czynności i wyznaczenie kolejnych zależy wyłącznie od danych sterujących (atrybutów) dostarczonych przez pakiet MIND do danego klienta LWE. W tych sytuacjach przetwarzanie dokumentu składowego jest zupełnie niezależne od stanów lokalnych innych dokumentów składowych. W rozprawie wyróżniłam cztery wzorce koordynacji dokumentu należące do tej kategorii: *sekwencyjny*, *rozdzielający*, *łączący* i *iterator*.

Wzorzec *sekwencyjny* został zdefiniowany na rysunku 5.4. Umożliwia on pracownikowi wiedzy przesłanie dokumentu do innego pojedynczego pracownika w całości lub w częściach. Rozdzielenie dokumentu na części może wymagać analizy jego wewnętrznej struktury, odpowiadającej formatowi jego zawartości i dokonania pewnej transformacji. Przykładowo, dla dokumentów XML może

to być transformacja XSLT, dla sformatowanego tekstu jakiś skrypt analizujący jego składnie (np. skrypt PERL), a dla innych formatów jakaś dedykowana aplikacja. Zatem w celu poprawnego odebrania i scalenia rozdzielonego dokumentu przez klienta LWE odbiorcy, dokument powinien przynieść ze sobą potrzebną do tego funkcjonalność.



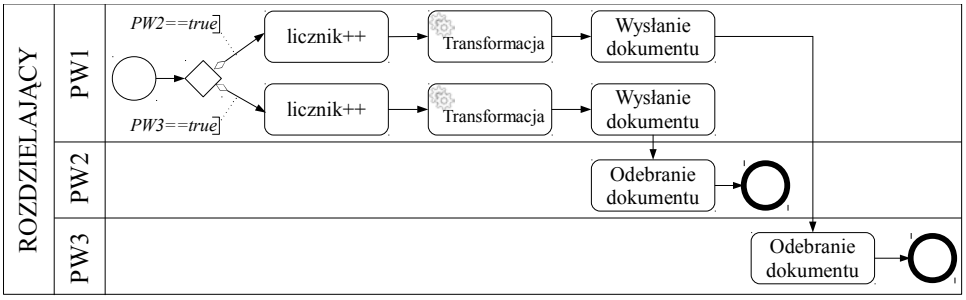
Rys. 5.4. Wzorzec sekwencyjnego przepływ dokumentu

Ponieważ rodzaj transformacji zależy od struktury i formatu dokumentu, we wszystkich zaprezentowanych wzorcach, funkcjonalność przyniesiona przez dokument w celu przekształcenia jego treści jest oznaczona ogólnie, jako czynność o nazwie Transformacja. W zależności od konkretnej implementacji może być ona wykonywana manualnie przez pracownika wiedzy lub automatycznie poprzez uruchomienie odpowiednich usług.

We wzorcu sekwencyjnym, dokument może zostać wysłany w kilku osobnych częściach, wyodrębnionych przez transformację. Każdej z tych części przypisywany jest kolejny numer, zaś ostatnia z wysyłanych części oznaczana jest dodatkowo znacznikiem końca (ang. *sentinel*). Dzięki temu, przy odbieraniu wiadomości w dowolnej kolejności, klient LWE danego odbiorcy może określić, czy wpłynęły już wszystkie części dokumentu. Przykładowo, pracownik PW2 odebrał części dokumentu posiadające numery: 1, 2 i 5, przy czym dokument 5 ma przypisaną wartość atrybutu znacznik_końca = true. W tym momencie wiadomo, że PW1 wysłał już wszystkie części dokumentu, że ich liczba wynosi 5 i należy poczekać jeszcze na przybycie części o numerach 3 i 4, po czym będzie można wykonać transformację łączącą i zakończyć wykonywanie danego wzorca.

Wzorzec sekwencyjny, w którym licznik > 1 odpowiada wzorcowi *rozdzielania i łączenia wątków* (tabela 3.2, wiersz 11.). W przeciwnym przypadku wysyłana jest tylko pojedyncza wiadomość do konkretnego odbiorcy, co odpowiada wzorcowi *sekwencji* (tabela 3.2, wiersz 1.).

Kolejnym wzorcem koordynacji dokumentu jest wzorzec *rozdzielający*, przedstawiony na rysunku 5.5. Wątek przepływu dokumentu jest rozdzielany na kilka równoległych wątków, których liczba jest ustalana na podstawie tranzycji.



Rys. 5.5. Wzorzec rozdzielający dokument

W zależności od funkcjonalności wykonanej w Transformacji, wzorzec rozdzielający może występować w wersji *kopiującej* lub *rozdzielającej* zawartość dokumentów składowych. Wersja kopiująca tworzy identyczne kopie dokumentów składowych, przez co każdy z wyznaczonych odbiorców otrzyma ten sam dokument, zaś wersja rozdzielająca dzieli zawartość dokumentu składowego na mniejsze części, zatem wskazani odbiorcy otrzymują tylko dedykowane im fragmenty dokumentu składowego.

Ewentualne rozdzielenie dotyczy plików z treścią dokumentu MIND, bowiem pliki zawierające opis struktury logicznej dokumentu (zob. podrozdział 4.3) są zazwyczaj kopiowane w punkcie rozdzielenia, a wartość atrybutu `CURRENT_ACTIVITY` każdej kopii komponentu przepływu pracy zostaje zamieniona na identyfikator następnej czynności, wyznaczonej zgodnie z ustaloną ścieżką przepływu. Przykładowo, gdyby w przepływie pracy zamieszczonym na rysunku 5.5 zostały spełnione oba warunki: PW2 == true i PW3 == true, to niezależnie od transformacji dokumentu składowego, powstaną dwie kopie zasobów dokumentu MIND: jedna kopia, zgodnie ze spełnieniem warunku PW2 == true, trafi do pracownika PW2 i atrybut `CURRENT_ACTIVITY` będzie wskazywał czynność Odebranie dokumentu umieszczoną na torze tego pracownika; druga kopia, zgodnie z warunkiem PW3 == true, zostanie wysłana do pracownika PW3 z odpowiednio ustalonym atrybutem `CURRENT_ACTIVITY` wskazującym czynność Odebranie dokumentu umieszczoną na jego torze.

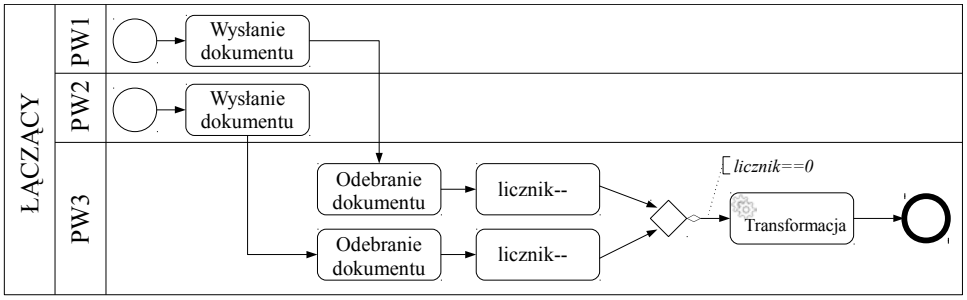
W przypadku bardzo rozbudowanych procesów można również ustalić dodatkową politykę rozdzielania plików zawierających opis struktury logicznej dokumentu. Biorąc pod uwagę, że są to dokumenty XML, można wykorzystać do tego celu transformację XSLT [AMRZ01]. Przesyłanie komponentu przepływu pracy w całości ma jednak tę zaletę, że każdy dokument przynosi do klienta LWE informację o strukturze i uczestnikach całego procesu.

Liczba kopii pakietu MIND, które powstają we wzorcu rozdzielającym, może być ustalona w momencie projektowania komponentu przepływu pracy (wówczas stosowane są tranzycje bezwarunkowe) lub też wyznaczana dynamicznie w punkcie

rozdzielenia na podstawie tranzycji warunkowych. Niezależnie od sposobu ustalania dalszego wątku przepływu, liczba utworzonych kopii jest zapisywana jako wartość atrybutu licznik w każdej kopii komponentu przepływu pracy. Jest to niezbędna informacja w przypadku konieczności łączenia rozdzielonych dokumentów, co zostanie wyjaśnione przy omawianiu kolejnego wzorca.

Wzorzec rozdzielający odpowiada kilku wzorcom wymienionym w tabeli 3.2: sytuacja, w której zastosowano tranzycje bezwarunkowe lub gdy wszystkie warunki zwróciły wartość true, odpowiada *rozbiciu równoległemu* (wiersz 2.); jeśli na podstawie analizy warunków została wybrana tylko jedna tranzycja, jest to wzorzec *wyłącznego wyboru* (wiersz 4.); natomiast pozostałe sytuacje odnoszą się do wzorca *wielokrotnego wyboru* (wiersz 6.).

Wzorzec *łączący* dokument, przedstawiony na rysunku 5.6, stanowi dopełnienie opisanego poprzednio wzorca rozdzielającego. Każdy komponent przepływu pracy zawiera atrybut licznik określający liczbę dokumentów powstałych w punkcie rozdzielenia. Ta wartość jest odczytywana przez klienta LWE na podstawie analizy pierwszego z przybyłych pakietów MIND i przypisywana zmiennej lokalnej licznik tego klienta LWE. Wartość tej zmiennej lokalnej jest zmniejszana o jeden w ramach czynności licznik-- z każdym przybyciem kolejnego pakietu MIND. Wzorzec łączący oczekuje na kolejne pakiety MIND, dopóki wartość zmiennej lokalnej licznik jest większa od zera. Każdy z otrzymanych dokumentów może być od razu przetwarzany i przesyłany dalej lub być zapisywany w odpowiednim katalogu (buforze) i przetwarzany dopiero po dostarczeniu wszystkich oczekiwanych dokumentów. Pierwsza sytuacja odpowiada wówczas wzorcowi *wielokrotnego łączenia* (tabela 3.2, wiersz 8.), zaś druga sytuacja odpowiada, w zależności od liczby oczekiwanych dokumentów, wzorcom *synchronizacji* (tabela 3.2, wiersz 3.) lub *zsynchronizowanemu łączeniu* (tabela 3.2, wiersz 7.). W obu tych sytuacjach, wzorzec zostaje zrealizowany, gdy wartość zmiennej lokalnej licznik wyniesie 0.



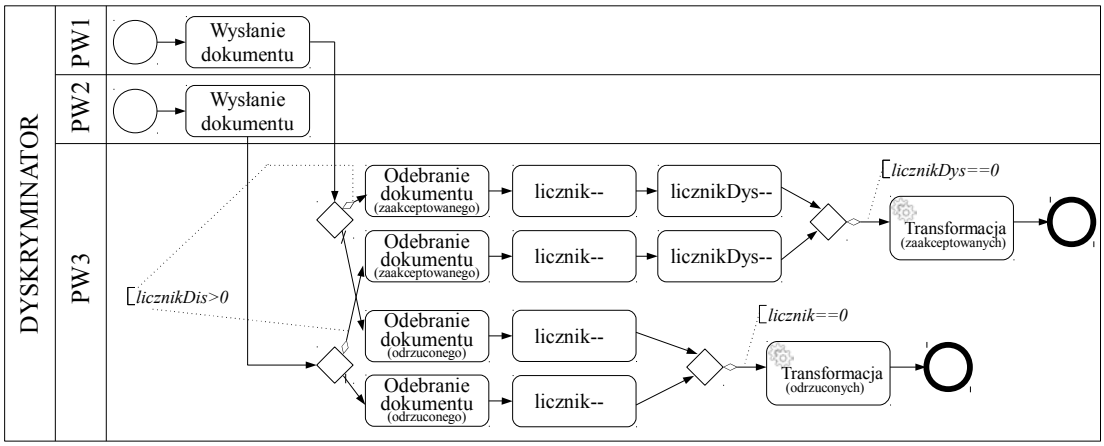
Rys. 5.6. Wzorzec łączący dokument

Scalenie różnych wersji dokumentów umieszczonych w buforze wymaga użycia funkcjonalności, która jest zależna od sposobu ich rozdzielenia, a więc od tego, czy

dokument składowy został skopiowany czy rozdzielony na części. Jeśli dokument został rozdzielony na niezależne fragmenty, jego scalenie może oznaczać prostą konkatenację jego części. Łączenie treści może być też wykonywane manualnie przez pracownika wiedzy. Zaś automatyczne łączenie skomplikowanej zawartości może wymagać bardzo złożonych funkcji [OUMI05, OUM$^+$05, LR05].

Jeśli odebrany dokument zawiera atrybut licznik = 1, wówczas wzorzec łączący odpowiada wzorcowi *prostego łączenia* (tabela 3.2, wiersz 5.), nie wymaga on wówczas scalania treści dokumentu i może od razu kontynuować proces.

Specjalnym przypadkiem wzorca łączącego jest *dyskryminujące łączenie* dokumentu (dyskryminator) zaprezentowany na rysunku 5.7. Wzorzec ten oczekuje na pewną liczbę dokumentów, które zostaną zaakceptowane i po dokonaniu pewnej transformacji, będą mogły kontynuować proces. Liczba ta jest wskazana przez wartość atrybutu licznikDys, która jest przypisana zmiennej lokalnej klienta LWE na podstawie analizy pierwszego z dostarczonych pakietów MIND lub jest wprowadzana przez pracownika. Każde przybycie zaakceptowanego dokumentu zmniejsza wartość zmiennej lokalnej licznikDys o jeden (aż do zera). Po akceptacji dokumentów, gdy zmienna licznikDys = 0, wzorzec nie kończy swojego działania, tylko oczekuje na pozostałe dokumenty, które zostaną odrzucone. Liczba wszystkich dokumentów jest określona przez atrybut licznik, zatem liczba dokumentów, które należy odrzucić, aby zakończyć wykonywanie tego wzorca, jest równa różnicy wartości atrybutów licznik-licznikDys.

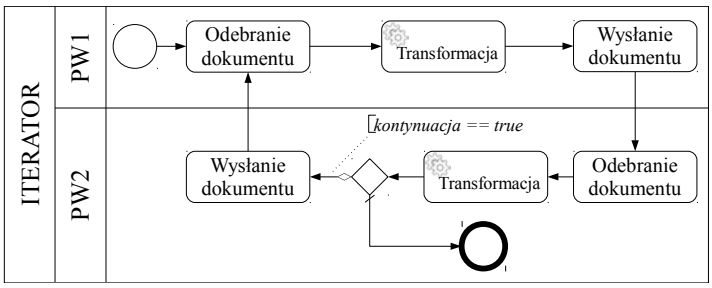


Rys. 5.7. Wzorzec dyskryminującego łączenia dokumentu

W sytuacji, gdy licznikDys < licznik wzorzec dyskryminującego łączenia odpowiada wzorcowi dyskryminatora (tabela 3.2, wiersz 9.). W przeciwnym przypadku nie są odrzucane żadne dokumenty, zatem jest to wzorzec łączący dokument przedstawiony powyżej.

Iterator dokumentu przedstawiony na rysunku 5.8 pozwalana na powtarzanie

wykonywania pewnej sekwencji czynności, kontrolowane przez czynność „bramkę", w której sprawdzany jest warunek wyjścia z pętli. Ogólny zapis kontynuacja == true oznacza pewną funkcjonalność przyniesioną przez dokument, która na podstawie zawartości dokumentu, wskazuje klientowi LWE wybór odpowiedniej tranzycji: kontynuującej lub kończącej pętle.



Rys. 5.8. Wzorzec iteratora dokumentu

Rysunek 5.8 prezentuje przykład pętli strukturalnej, która posiada jeden punkt wejściowy i wyjściowy oraz warunek rozpatrywany na wyjściu. Jednakże sam mechanizm sprawdzania warunków na podstawie danych przyniesionych przez dokument i wybieranie odpowiednich tranzycji, pozwala na konstrukcję zarówno pętli strukturalnych (tabela 3.2, wiersz 18.), jak i dowolnych cykli, które mają więcej niż jeden punkt wejścia lub wyjścia (tabela 3.2, wiersz 17.).

Szczególnym przypadkiem użycia wzorca iteracyjnego jest przesyłanie dokumentu przez pracownika wiedzy do siebie samego. Wówczas iterator jest wykonywany jako podproces w obrębie jednej czynności przypisanej do danego pracownika. Jeśli wiadomości są przetwarzane w kolejności zgodnej z ich dostarczeniem, czyli są zorganizowane w strukturze kolejki (FIFO), wówczas ma miejsce wzorzec strukturalnej pętli. W przeciwnym przypadku, gdy wiadomości są odczytywane w odwrotnej kolejności od dostarczenia, tak jak w przypadku stosu (LIFO), wówczas iterator dokumentu odpowiada wzorcowi rekurencji (tabela 3.2, wiersz 19.).

Iterator dokumentu może być także użyty w połączeniu z innymi wzorcami do implementacji bardziej skomplikowanych wzorców wymienionych w podrozdziale 3.2.2. Jednym z nich jest *przeplatany obieg równoległy* (tabela 3.2, wiersz 14.), który może być skonstruowany jako kombinacja wzorców koordynacji dokumentu: rozdzielającego (w wersji wyłącznego wyboru), sekwencyjnego oraz łączącego (w wersji prostego łączenia). Innym wzorcem jest *uogólnione łączenie typu AND* (tabela 3.2, wiersz 10.), który stanowi w zasadzie kombinację iteratora i wzorca łączącego dokument. Ta redukcja kanonicznego zbioru wzorców z podrozdziału 3.2.2 jest możliwa dzięki temu, że komponent przepływu pracy wykorzystuje mechanizm definiowania dodatkowych atrybutów XPDL do sterowania lokalnym

stanem klientów LWE i wprowadza pojęcie stanu do ścieżki przepływu dokumentu MIND, które oryginalnie nie zostało uwzględnione w XPDL [RHAM06].

5.3.2. Wzorce ze sprzężonym stanem dokumentu

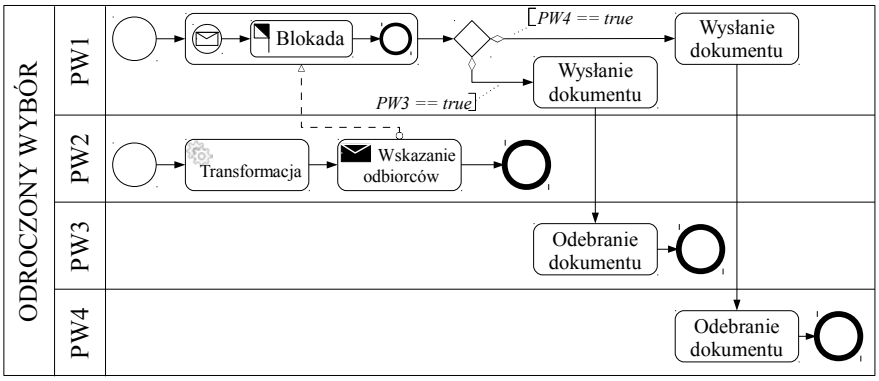
W kategorii wzorców z rozproszonym stanem dokumentu omówionych w poprzednim podrozdziale, głównym źródłem informacji dla klienta LWE pozwalającym na wyznaczenie kolejnych czynności był odebrany dokument, a dokładniej wartości odpowiednich atrybutów dołączonych do jego komponentu przepływu pracy. Jednakże, w niektórych sytuacjach, klient LWE może potrzebować informacji o stanach innych klientów LWE, w celu prawidłowej realizacji ścieżki migracji dokumentu. Te sytuacje tworzą kategorię wzorców ze sprzężonym stanem dokumentu, które wymagają wprowadzenia mechanizmu asynchronicznych sygnałów, odbieranych przez klienty LWE z innej części procesu.

Sprzężenie stanów poszczególnych klientów LWE nie jest standardowe w przypadku rozproszonego zarządzania przepływem pracy z wykorzystaniem komunikacji za pośrednictwem poczty elektronicznej i bez żadnej centralnej jednostki sterującej. Wymaga zatem wprowadzenia mechanizmu przesyłania dodatkowych sygnałów. Mechanizm ten może być zaimplementowany na wiele sposobów, np. w oparciu o wiadomości SMS, połączenia telefoniczne, komunikatory internetowe czy też niezwiązane z przepływem MIND wiadomości poczty elektronicznej. Czynność, której wykonanie jest zależne od stanu innego klienta LWE, powinna zostać zablokowana na danym urządzeniu, dopóki nie zostanie odebrany sygnał pozwalający na jej realizację. Taka blokada jest zaimplementowana w MIND jako semafor binarny (zob. rys. 5.3).

Pierwszym wzorcem należącym do bieżącej kategorii wzorców jest *odroczony wybór* (ang. *deffered choice*), przedstawiony na rysunku 5.9. Prezentuje on sytuacje, w której wyznaczenie następnych czynności i wysłanie dokumentu jest odroczone, dopóki do danego klienta LWE nie wpłynie sygnał z informacją pozwalająca na wybór konkretnej tranzycji.

Dokument po odebraniu go przez PW1, ma początkowo wartość atrybutu-semafora Blokada = true. Jeśli został odebrany przez PW1 przed wpłynięciem sygnału od PW2, klient LWE pozostawia go w odpowiednim katalogu i w czasie oczekiwania na sygnał może pośredniczyć w przetwarzaniu innych dokumentów MIND. Po otrzymaniu sygnału od PW2, przetwarzanie dokumentu zostaje wznowione. Może się też zdarzyć, że sygnał od PW2 wpłynie przed odebraniem dokumentu. Wówczas jest przechowywany w pamięci klienta LWE i pozwala na przetworzenie dokumentu zaraz po jego odebraniu.

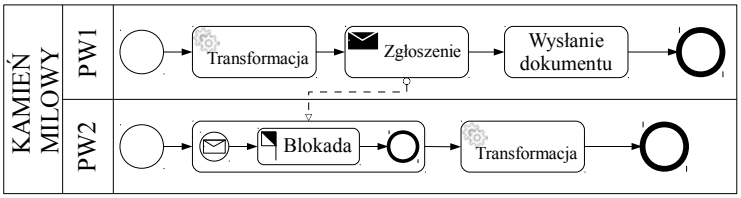
W przypadku wzorca odroczonego wyboru, sygnał przynosi do klienta LWE dane,



Rys. 5.9. Wzorzec odroczonego wyboru

na podstawie których zostają wybrane odpowiednie tranzycje i dokument zostaje wysłany do wskazanych odbiorców. Na rysunku 5.9 odbiorcami dokumentu mogą być pracownicy PW3 i PW4.

Kolejny wzorzec – *kamień milowy* (ang. *milestone*), przedstawiony na rysunku 5.10, nie wymaga w swojej implementacji, by wraz z sygnałem przesyłane były dodatkowe dane. Czynność blokująca wstrzymuje przepływ dokumentu u pracownika PW2, dopóki nie zostanie wykonana czynność Zgłoszenie przypisana pracownikowi PW1 i polegająca na wysłaniu sygnału zmieniającego ustawienie semafora na Blokada = false. Tak samo, jak w poprzednim wzorcu, sygnał ten może być odebrany przez klienta LWE zarówno przed, jak i po wpłynięciu dokumentu.

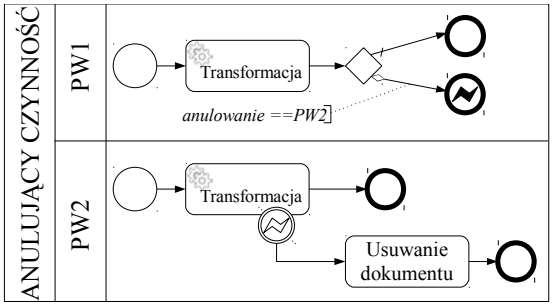


Rys. 5.10. Wzorzec kamienia milowego

Wzorce: *odroczony wybór* i *kamień milowy*, które wykorzystują czynność blokującą i pozwalającą na odbieranie zewnętrznych sygnałów, odpowiadają wzorcom umieszczonym odpowiednio w wierszach nr 13 i 15 tabeli 3.2.

Niestety, implementacja wzorców anulujących i kończących, które razem zostały umieszczone w wierszu 16 tabeli 3.2, jest bardziej skomplikowana w przypadku rozproszonego zarządzania przepływem pracy w oparciu o pocztę elektroniczną. Wymagają one bowiem ustalenia lokalizacji wykonywanych czynności oraz zdefiniowania sposobu realizacji anulowania czynności w odniesieniu do treści dokumentu. Rysunek 5.11 prezentuje wzorzec pozwalający na anulowanie czynności procesu.

Po wykonaniu pewnych transformacji na dokumencie przez pracownika PW1



Rys. 5.11. Wzorzec anulujący czynność procesu

może się okazać, że czynność wykonywana przez innego pracownika PW2 musi zostać anulowana, a związany z nią dokument musi zostać usunięty bądź wykonane już w nim zmiany wycofane (operacja typu „undo"). W celu anulowania czynności, wysyłany jest specjalny sygnał anulujący, który powinien być przechwycony przez odpowiednią czynność. Ponieważ nie ma centralnej jednostki sterującej przepływem pracy, zatem należy uwzględnić trzy przypadki związane ze stanem realizacji danej czynności:

– dokument mógł jeszcze nie dotrzeć do pracownika PW2, a więc wyznaczona do anulowania czynność nie została jeszcze rozpoczęta, wówczas sygnał anulujący powinien być przechowany w pamięci klienta LWE i spowodować usunięcie dokumentu zaraz po jego dostarczeniu;

– dokument został już odebrany przez PW2 i jest w trakcie przetwarzania w czasie odebrania sygnału anulującego. W tym przypadku usunięcie dokumentu może być problematyczne, gdyż PW2 mógł wykonać już znaczną część pracy i dokonać wielu zmian w dokumencie, a także przesłać dalej gotowe części dokumentu lub zaangażować pewne zewnętrzne usługi. Wymagany jest wówczas bardziej złożony mechanizm wycofywania zmian opracowany dla każdej czynności komponentu przepływu pracy. Można zauważyć, że w przypadku anulowania czynności, nie jest potrzebny semafor, gdyż decyzja o anulowaniu czynności jest podejmowana natychmiastowo po odebraniu sygnału przez klienta LWE.

– dana czynność została już zakończona, praca na dokumencie wykonana, a dokument został przesłany do kolejnego pracownika, tak więc anulowanie tej konkretnej czynności nie ma już sensu i nie przyniesie żadnego efektu.

Zatem, anulując konkretną czynność w procesie, nie ma gwarancji usunięcia dokumentu, z którym jest ona związana. Alternatywą jest wysłanie sygnału anulującego do wszystkich pracowników wiedzy związanych z danym procesem,

których lista znajduje się w komponencie pracowników wiedzy (zob. podrozdział 4.3.3) lub do pracowników związanych z czynnościami na całej ścieżce przepływu danego dokumentu składowego, która jest dostępna w komponencie przepływu pracy. Jest to wówczas implementacja wzorców *anulowania sprawy* lub *anulowania części sprawy* (zob. podrozdział 3.2.2). Rozwiązanie to jest wykonalne, choć kosztowne i nadmiarowe. Dodatkowe komplikacje mogą się pojawić, gdy w trakcie procesu będą dodawani nowi autorzy i kolejne czynności nieznane w części procesu, z której pochodzi sygnał. Zatem sygnał anulujący musiałby trafić do wszystkich czynności procesu, łącznie z podprocesami i czynnościami dodanymi w trakcie jego realizacji.

Wzorzec anulujący czynność procesu przedstawiony na rysunku 5.11 odnosi się także do wzorca *jawnego zakończenia* (zob. podrozdział 3.2.2), z tą różnicą, że sygnał anulujący jest wysyłany do czynności procesowych, gdy proces uaktywni specjalnie wyróżnioną czynność kończącą.

Natomiast wzorzec *niejawnego zakończenia* jest oznaczany przez niewysyłającą żadnych sygnałów czynność zdarzenia kończącego (zob. rys. 5.3) i dotyczy punktu w procesie nadrzędnym[1], z którym nie są związane już żadne tranzycje wyjściowe.

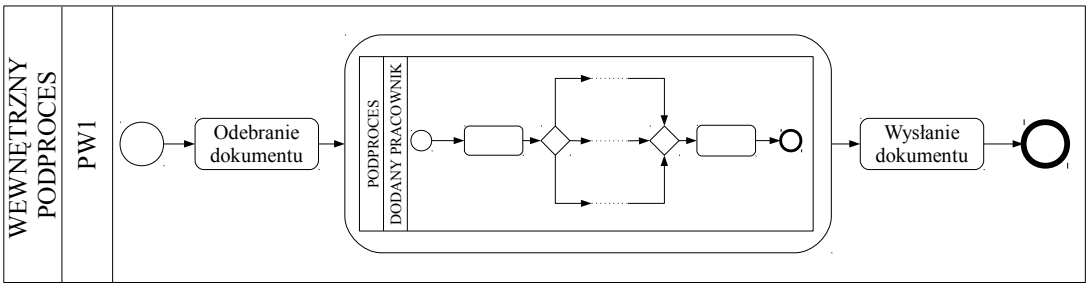
5.3.3. Wzorce z zagnieżdżonym stanem dokumentu

W komponencie przepływu pracy mogą znajdować się czynności zawierające pewien zestaw innych, zagnieżdżonych czynności wykonywanych jako podproces. Podprocesy mogą definiować czynności nie ujęte w początkowej strukturze dokumentu, dodawane w trakcie realizacji procesu z udziałem nowych pracowników. Zatem w przypadku podprocesów można mówić o stanach zagnieżdżonym w lokalnym stanie dokumentu (zob. podrozdział 2.4).

Podprocesy realizowane przez organizacje można podzielić na *wewnętrzne* i *zewnętrzne*. *Podproces wewnętrzny*, przedstawiony na rysunku 5.12, jest dodawany do dokumentu podczas realizacji jego migracji przez pracownika posiadającego uprawnienia do edycji komponentu przepływu pracy. Zarówno struktura podprocesu, jak i dodani pracownicy są znani pracownikowi definiującemu podproces.

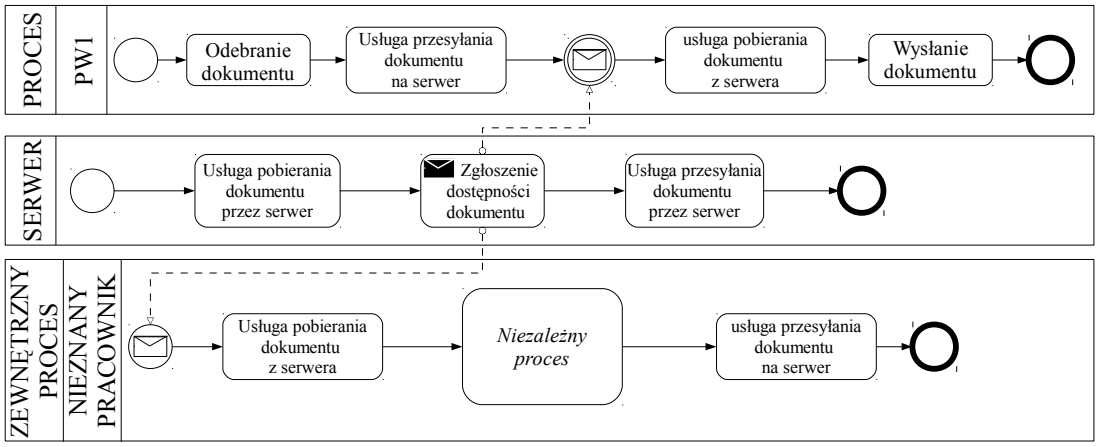
Podproces zewnętrzny, przedstawiony na rysunku 5.13, jest związany z wykonaniem pewnego zadania przez proces, którego zarówno struktura, jak i uczestnicy nie są znani zlecającemu go pracownikowi wiedzy. W wymianie dokumentów pomiędzy procesem głównym a podprocesem zewnętrznym pośredniczy pewien serwer, udostępniający usługi przesyłania dokumentu na serwer (ang. *upload*) i pobierania

[1]Czynność zdarzenia kończącego wykonana w podprocesie powoduje zakończenie podprocesu i kontynuowanie wykonywania procesu o jeden poziom zagnieżdżenia wyżej.



Rys. 5.12. Wzorzec wewnętrznego podprocesu

dokumentów z serwera (ang. *download*). Może też, dodatkowo, przesyłać komunikaty o dostępności dokumentu wskazanemu odbiorcy. Podproces zewnętrzny rozpoczyna się więc poprzez uruchomienie zewnętrznej usługi, która umożliwia przesłanie dokumentu na serwer. Serwer informuje o dostępności dokumentu wykonawców zewnętrznego procesu, którzy go pobierają i wykonują na nim zleconą pracę. Po jej wykonaniu, umieszczają gotowy dokument na serwerze. Serwer powiadamia zleceniodawcę o dostępności dokumentu i umożliwia jego pobranie, po czym może być kontynuowany pierwotny procesu. Zewnętrzny podproces nie musi być realizowany przez aplikację opartą na architekturze MIND.



Rys. 5.13. Wzorzec zewnętrznego podprocesu

Wzorce z zagnieżdżonym stanem dokumentu, przedstawione powyżej, stanowią uogólnienie wzorca zwielokrotnionego wykonania czynności (tabela 3.2, wiersz 12.).

Wzorce koordynacji dokumentu są elementami, z których można skonstruować dowolny proces przepływu dokumentów w organizacjach opartych na wiedzy. Umożliwiają one zatem realizację procesów decyzyjnych w trybie obliczeń zespołowych, wykonywanych w środowisku rozproszonym. Mogą być one bowiem zrealizowane w ramach rozproszonej usługi zarządzającej przepływem pracy składającej się z klientów LWE, które umożliwiają pracę na różnych urządzeniach, w tym urzą-

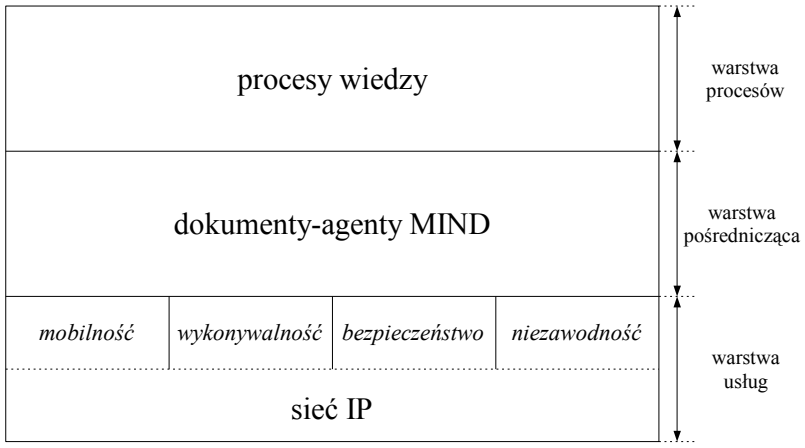
dzeniach mobilnych oraz wymagają niewielkiej komunikacji z Internetem i miedzy sobą.

Ponadto klienty LWE mogą być zaimplementowane z wykorzystaniem dowolnych technologii, które umożliwiają realizację architektury MIND. Dokument MIND jest bowiem niezależny od implementacji klientów LWE. Pozwala to na przekazywanie dokumentów MIND również pomiędzy różnymi organizacjami, które posiadają klientów LWE zaimplementowanych w odmienny sposób.

Rozdział 6

Walidacja modelu otwartej architektury rozproszonego dokumentu elektronicznego

Architektura dokumentu MIND pozwala na implementację otwartego systemu agentowego zgodnie z modelem architektury 3-warstwowej, w którym dokumenty-agenty MIND stanowią warstwę pośredniczącą (rys. 6.1).



Rys. 6.1. Architektura 3-warstwowa systemu opartego na modelu MIND

Warstwa usług dostarcza podstawowe funkcjonalności, niezbędne do realizacji proaktywnych dokumentów-agentów, których architekturę zaproponowałam w rozdziale 4. Kluczowymi z punktu widzenia tezy rozprawy są *wykonywalność* i *mobilność*. Przekształcenie dokumentu z postaci statycznej (spakowanej), w jakiej znajduje się podczas przesyłania między węzłami pracowników organizacji, do postaci dynamicznej (wykonywalnej) jest możliwe dzięki wykorzystaniu ogólnie dostępnych narzędzi do wiązania statycznych danych zawartych w pakiecie MIND z dynamicz-

nymi obiektami (komponentami MIND).

Po takim przekształceniu, dokument uzyskuje zdolność samodzielnego wykonywania operacji, zarówno wbudowanych metod jego poszczególnych obiektów, jak i zewnętrznych względem niego lokalnych usług stacji, na której wykonywane są wspomniane operacje lub zewnętrznych usług dostępnych w sieci. Na rysunku 6.1 te lokalne i zewnętrzne usługi zostały zaznaczone łącznie w warstwie usług systemu agentowego. Z kolei usługi umożliwiające mobilność agentów MIND zapewniają transport spakowanych obiektów MIND między urządzeniami pracowników wiedzy i również są dostarczane przez warstwę usług środowiska sieciowego organizacji. W dalszej części rozdziału, omówię alternatywne rozwiązania usług transportowych, jakie zastosowałam w prototypowych realizacjach architektury MIND.

Usługi *bezpieczeństwa* i *niezawodności*, choć istotne w pracy grupowej w systemie rozproszonym, nie mają bezpośredniego związku z tezą rozprawy i zostały przeze mnie pominięte. Badania w tym zakresie aktualnie prowadzę w ramach projektu NCN [MeN13].

Warstwa pośrednicząca agentów MIND umożliwia realizację procesów wiedzy w organizacjach ludzkich (opisanych szczegółowo rozdziale 2), w postaci przepływów pracy dokumentów pomiędzy pracownikami wiedzy. W dalszej części rozdziału argumentuję, że agenty MIND zapewniają możliwość realizacji kanonicznego zbioru wzorców koordynacji dokumentów, zdefiniowanego w rozdziale 5. Na tej podstawie wykażę tezę rozprawy – ogólne podejście procesowe zaproponowane przez zespół van der Aalsta [ARH11] opiera się na zbiorze kanonicznych wzorców przepływu pracy, występujących w organizacjach ludzkich, realizujących niealgorytmiczne procesy wiedzy zgodnie z modelem przedstawionym na rysunkach 2.1 i 3.1. Zatem kanoniczny zbiór wzorców koordynacji dokumentów, zdefiniowany przeze mnie w podrozdziale 5.3 na podstawie wspomnianego zbioru wzorców ogólnego podejścia procesowego, umożliwia zarządzanie wiedzą w takich organizacjach.

Z powyższego rozumowania wynika, że kluczem do udowodnienia tezy rozprawy jest wykazanie *realizowalności* zarówno architektury MIND w oparciu o dostępne usługi sieciowe organizacji jak i wzorców zbioru kanonicznego w oparciu o dokumenty-agenty MIND.

Jak wykażę dalej, realizacja architektury MIND nie jest zależna od konkretnych technologii i charakteryzuje się trwałością, rozumianą jako zdolność do funkcjonowania w środowiskach i przy zastosowaniu technologii, jakie mogą się pojawić dopiero w przyszłości (ang. *forward compatibility*). Pozwala to na dostosowanie jej do wymogów danej organizacji, a także na wymianę informacji między organizacjami. Implementacja w oparciu o istniejące i sprawdzone technologie eliminuje też konieczność tworzenia nowych technologii, które wymagałyby dodatkowej, czasochłonnej wali-

dacji.

Walidację architektury MIND przeprowadziłam w dwóch etapach związanych z rolami, jakie pełni warstwa pośrednicząca dokumentów-agentów MIND w 3-warstwowej strukturze systemu zarządzającego obiegiem dokumentów w organizacji względem warstwy usług oraz względem warstwy procesów:

1. mobilność i funkcjonalność agentów MIND została zaimplementowana alternatywnie w oparciu o dojrzałą platformę agentową JADE oraz luźno powiązany system (ang. *loosely coupled system*), wykorzystujący pocztę elektroniczną jako warstwę transportową,

2. implementowalność wzorców koordynacji dokumentu w architekturze MIND została zweryfikowana z wykorzystaniem istniejącego, centralnego systemu zarządzającego przepływem pracy Enhydra Shark, a następnie zademonstrowana w konkretnym studium przypadku – złożonym procesie zarządzania dużym przedmiotem w szkole wyższej.

6.1. Implementacja mobilności i funkcjonalności agentów MIND

Komponenty MIND, zgodnie z założeniami przedstawionymi w cyklu życia dokumentu (rys. 1.2, str. 19), mają cechy mobilnych agentów, tzn. zdolność samodzielnej migracji pomiędzy użytkownikami i usługami w systemie rozproszonym oraz autonomicznego wykonywania pewnych zadań, np. dokonywania serializacji i deserializacji, reorganizowania własnej struktury logicznej czy automatycznego aktualizowania swojej treści informacyjnej. Dla zapewnienia agentom możliwości działania, niezbędny jest pewien system agentowy, dostarczający środowisko do aktywacji agenta i umożliwieniu mu wykonywania swojej funkcjonalności na rozproszonych lokalizacjach oraz warstwę transportową do samodzielnej migracji agentów między różnymi urządzeniami.

Architektura MIND została zrealizowana alternatywnie w postaci dwóch systemów agentowych:

1. wykorzystującym istniejącą platformę agentową JADE,

2. opartym na poczcie elektronicznej jako powszechnie akceptowanej warstwie wymiany informacji i specjalnie zaprojektowanym kliencie poczty LWE.

Pierwsza z wymienionych implementacji wymagała dokonania wyboru systemu agentowego spośród wielu dostępnych platform. Ze względu na to, że w niektórych

zastosowaniach architektury MIND liczba komponentów może być znaczna (np. sprawa sądowa), wybór konkretnej platformy agentowej wymagał szczególnego rozważenia zagadnienia skalowalności z punktu widzenia przekazu komunikatów między agentami oraz akcji związanych z migracją agenta między węzłami systemu.

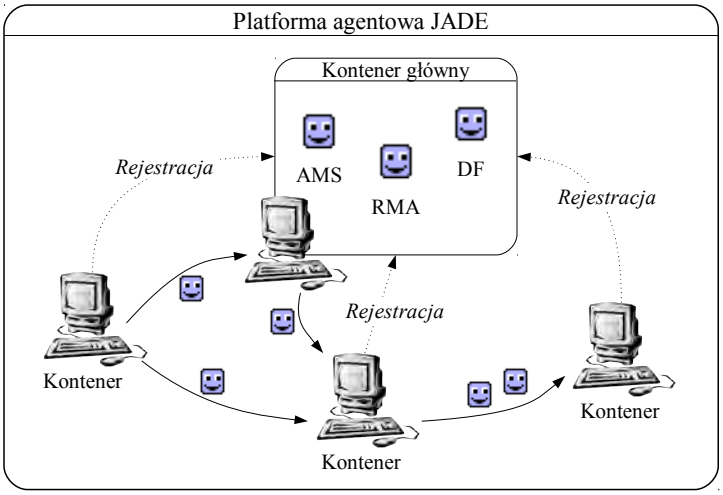
Badania przeprowadzone dla najpopularniejszych platform agentowych [TIM07], obejmujących *Aglets*, *Voyager*, *Grasshopper*, *Tryllian*, *JADE*, *Tracy* i *SPRINGS* wskazały, że dla aplikacji zawierających nie więcej niż 100 aktywnych agentów i wykonujących do kilkuset akcji polegających na przemieszczaniu się między węzłami i wysyłaniu komunikatów, zaobserwowano tylko nieznaczne różnice w wydajności, tym mniejsze im rzadziej agenty się komunikowały. Z punktu widzenia implementowalności architektury MIND wynik ten był pozytywny, gdyż zasadnicza aktywność każdego komponentu dokumentu MIND koncentruje się na migracji między stacjami i interakcją z użytkownikiem tylko tej stacji roboczej, na której aktualnie przebywa. Problemy z wydajnością modelu przetwarzania rozproszonego zorientowanego na dokumenty z wykorzystaniem platformy agentowej mogą ewentualnie wynikać z wielkości przesyłanych pakietów MIND.

Istotnym zagadnieniem z perspektywy wyboru istniejącej platformy agentowej jako podstawy implementacji MIND była też jej trwałość i zgodność ze specyfikacją standardu dla platform agentowych, opublikowaną przez FIPA (*Foundation for Physical Intelligent Agents*) [Fou02]. Z wymienionych wcześniej platform, najbardziej zbliżona do tego standardu okazała się platforma JADE [BCG07], rozwijana w formule wolnego oprogramowania od 1998 roku. Jak wykazano formalnie, mechanizmy dostarczane przez tę platformę gwarantują implementowalność standardowego zbioru wzorców zachowania agentów [TTOH01]. Należy podkreślić, że zbiór ten znacznie wykracza poza zbiór wzorców zachowania agentów zakładanych dla modelu przetwarzania zorientowanego na dokumenty przedstawionego w rozdziale 5.

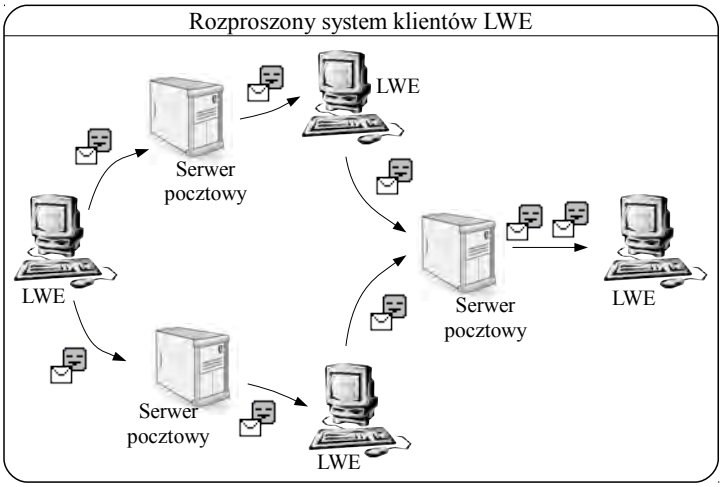
W drugiej z wymienionych implementacji, do realizacji mobilności komponentów MIND wykorzystałam stabilną i powszechnie używaną jako „warstwa transportowa" dokumentów pocztę elektroniczną. Do aktywacji agentów i realizacji wzorców koordynacji dokumentów niezbędne było zaprojektowanie aplikacji w formie „lekkiego" klienta LWE (rys. 5.1(b)), pozwalającego na odbieranie i wysyłanie pakietów MIND oraz na zamianę ich na dynamiczne obiekty, które mogą realizować swoją misję na lokalnym urządzeniu pracownika. Takie rozwiązanie pozwala uniezależnić system agentowy od konkretnej platformy i konieczności zarządzania nią, zaś poczta elektroniczna jest warstwą transportową dla dokumentów, umożliwiającą przesyłanie plików dowolnego typu i rozmiaru.

6.1.1. Mobilność dynamicznych komponentów dokumentu

Rysunek 6.2 przedstawia alternatywne sposoby realizacji mobilności dokumentów-agentów MIND.



(a) Platforma JADE



(b) Rozproszony system klientów LWE

Rys. 6.2. Model mobilności agentów w zależności od implementacji architektury MIND

Rysunek 6.2(a) prezentuje podstawowe elementy platformy JADE [BCG07], udostępniającej zbiór kontenerów, z których jeden pełni rolę *kontenera głównego*, uruchamiającego platformę. Kontener jest obiektem JADE uruchamianym na lokalnym komputerze i pozwalającym na aktywowanie agentów. Kontenery nie muszą być uruchamiane na oddzielnych komputerach – na jednym komputerze może działać kilka kontenerów. Każdy inny kontener musi nawiązać łączność z kontenerem głównym

w celu dołączenia do platformy JADE i może się połączyć tylko z jednym takim kontenerem.

W kontenerze głównym mogą być uruchomione agenty funkcyjne, sterujące platformą JADE i poruszającymi się w niej agentami. Agent *AMS* (ang. *Agent Management System*) to element systemu JADE, który zapewnia usługi nazewnicze (zapewnienia, by każdy agent miał unikatową nazwę) i umożliwia zdalne uruchamianie agentów. Agent *DF* (ang. *Directory Facilitator*) odpowiada za katalogowanie usług agentów (ang. *yellow pages*), każdy agent systemu może zgłaszać swoje usługi, dzięki czemu inne agenty mogą wyszukiwać w tym katalogu interesujące je usługi. Agent *RMA* (ang. *Remote Monitoring Agent*) umożliwia obserwacje zachowania agentów na innych stacjach roboczych, poprzez zapewnienie użytkownikowi graficznej reprezentacji aktualnego stanu platformy agentowej.

Komunikacja w systemie JADE jest realizowana przez usługę *Message Transport Service (MTS)*, która jest odpowiedzialna za przesyłanie agentów oraz wszelkich komunikatów pomiędzy agentami. Do komunikacji zewnętrznej, między platformami JADE służą protokoły *Message Transport Protocols (MTPs)* zgodne z definicją protokołów komunikacyjnych FIPA [Fou02]. Domyślnie są one oparte na protokołach HTTP [FGM+99] lub HTTPS [Res00]. Komunikacja wewnątrz platformy jest realizowana poprzez *Internal Message Transport Protocol (IMTP)*, który został zaimplementowany specjalnie dla platformy JADE i nie jest zgodny ze standardem FIPA. Podczas migracji, agent jest serializowany w kontenerze początkowym, przesyłany odpowiednim protokołem komunikacyjnym i deserializowany do postaci aktywnego agenta w kontenerze docelowym.

Rysunek 6.2(b) przedstawia model komunikacji w rozproszonym systemie klientów LWE. Klienty LWE są zainstalowane lokalnie na urządzeniach pracowników i nie są zarejestrowane w żadnej centralnej jednostce zarządzającej. Tworzą w ten sposób luźno powiązany (ang. *loosely coupled*) rozproszony system, w którym komunikacja odbywa się za pośrednictwem serwerów poczty elektronicznej. Części składowe dokumentu są przekazywane między klientami LWE w postaci spakowanego pakietu MIND, stanowiącego załącznik wiadomości elektronicznej. Wiadomość najpierw trafia na wskazany serwer pocztowy i LWE odbiorcy pobiera wiadomość z tego serwera. Odebranie dokumentu przez klienta LWE, powoduje aktywowanie pakietu MIND, czyli przekształcenie go z postaci statycznej w aktywną.

Komunikacja w systemie klientów LWE odbywa się zatem poprzez standardowe protokoły pocztowe. Protokół *Simple Mail Transfer Protocol (SMTP)* [Kle08] opisuje sposób przekazywania wiadomości z LWE na serwery pocztowe i pozwala na przekazywanie wiadomości utworzonych w standardzie MIME [FB96]. Zatem umożliwia przesyłanie wiadomości zawierających dowolny typ zawartości oraz

zdefiniowane dodatkowe nagłówki, pozwalające, przykładowo, na identyfikację poczty pochodzącej z LWE. Do odbierania wiadomości z serwerów pocztowych przez LWE wykorzystywany jest najbardziej popularny protokół *Post Office Protocol version 3 (POP3)* [MR96] lub zaprojektowany jako jego następca protokół *Internet Message Access Protocol (IMAP)* [Cri03], posiadający większe możliwości zarządzania listami wiadomości na zdalnym serwerze.

W obu przypadkach mogą pojawić się standardowe problemy komunikacyjne, które są związane z przesyłaniem wiadomości poprzez sieć publiczną, nie zaś z samą architekturą MIND. W przypadku komunikacji na platformie JADE może wystąpić problem blokowania portów HTTP lub IMTP przez system użytkownika, co wymaga przeprowadzenia odpowiedniej konfiguracji tzw. zapory ogniowej (ang. *firewall*). W przypadku komunikacji za pośrednictwem poczty elektronicznej, mogą pojawić się problemy związane z ochroną antyspamową, co wymaga odpowiedniej konfiguracji ustawień konta pocztowego użytkownika lub samego serwera pocztowego. Jednakże w organizacjach sieciowych, komunikacja za pośrednictwem poczty elektronicznej jest powszechnie stosowana, zatem ewentualne problemy z przekazywaniem wiadomości są w tym przypadku lepiej znane użytkownikom, niż konfiguracja platformy JADE.

6.1.2. Funkcjonalność dokumentów MIND

Zgodnie z cyklem życia dokumentu MIND, jego statyczna reprezentacja jest zamieniana na zbiór dynamicznych komponentów (agentów), które mogą realizować swoją misję, przebywając w lokalnym środowisku użytkownika (czyli w kontenerze JADE lub w katalogach roboczych klienta LWE). Środowisko to pozwala na aktywację agentów, które mogą autonomicznie wykonywać powierzone im zadania, takie jak interakcja z użytkownikiem i usługami, reorganizowanie własnej struktury logicznej oraz aktualizowanie swojej zawartości.

Do konwersji dokumentu XML na zbiór dynamicznych komponentów, zawierających jego dane i udostępniających zbiór operacji do działania na nich, służą specjalne narzędzia wiązania danych logicznych XML z obiektami (ang. *XML data binding*) w procesie deserializacji XML określanym jako *unmarshalling*. Proces ten polega na znalezieniu odpowiedniego odwzorowania szablonu struktury logicznej dokumentu XML (schematu) na strukturę obiektów. Na podstawie tego odwzorowania, narzędzia, o których mowa, tworzą wykonywalne obiekty, np. języka Java. Mogą one także dokonywać serializacji obiektów ponownie do postaci dokumentu XML, w procesie określanym jako *marshalling*. Popularne narzędzia wiązania danych realizują operacje *unmarshalling* i *marshalling* z pewnymi ograniczeniami, które w ogólności

dotyczą odwracalności operacji wiązania (ang. *round-tripping limitations*) i użyteczności (ang. *feature limitations*) [Bou11].

Problemy z odwracalnością operacji wiązania występują, gdy dokument otrzymany w wyniku deserializacji i ponownej serializacji różni się od dokumentu oryginalnego. Istniejące narzędzia na ogół radzą sobie dobrze z odwracalnością operacji wiązania w odniesieniu do elementów, atrybutów i tekstu oraz zachowaniem zależności wynikających z ich zagłębienia w strukturze logicznej dokumentu, gorzej natomiast wypadają w testach sprawdzających odtwarzanie pierwotnej kolejności elementów znajdujących się na tym samym poziomie zagnieżdżenia oraz zachowanie komentarzy, instrukcji przetwarzania języka skryptowego i przestrzeni nazw.

Ograniczona użyteczność narzędzi wiązania danych wynika najczęściej z faktu, że nie implementują one w całości standardowych mechanizmów deklaracyjnych XML Schema, takich jak mieszana zawartość elementu, wieloznaczniki (ang. *wild cards*) i ograniczenia dla typów złożonych. Znacznie poważniejszym ograniczeniem użyteczności jest fakt, iż praktycznie żadne znane mi narzędzie do wiązania danych nie potrafi przetwarzać dokumentu fragmentami, tzn. po wyodrębnieniu określonego fragmentu z dokumentu źródłowego dokonać jego konwersji na obiekty, a po ich wykorzystaniu ponownie przekonwertować na odpowiedni fragment, wstawiany do dokumentu oryginalnego bez zmiany jego pozostałych części. To graniczenie ma poważne konsekwencje dla realizacji przepływów pracy, w których cały dokument przechodzi od czynności do czynności realizowanych przez różne aplikacje na wybranych fragmentach dokumentu. W wypadku architektury MIND to ograniczenie nie stanowi jednak problemu, bowiem dokument dzielony jest na odpowiednie fragmenty jeszcze przed rozpoczęciem przetwarzania. Odpowiadające im obiekty podlegają następnie przetwarzaniu rozproszonemu przez aplikacje użytkowników związane z poszczególnymi czynnościami przepływu pracy bez ingerencji w całość dokumentu.

Jest bardzo wiele narzędzi, które realizują operację wiązania danych do obiektów różnych języków. Do implementacji architektury MIND, bazującej na języku Java, brałam pod uwagę trzy powszechnie dostępne narzędzia: *JAXB (Java Architecture for XML Binding)* [Ora13c], *XML Beans* [Apa04] i *JiBX* [Sos11].

Narzędzie JAXB służy do wiązania danych w oparciu o notację XML Schema i jest rozwijane przez korporację Oracle. Stanowi też część standardowej platformy Java (Java Standard Edition) od wersji 1.6, przez co nie wymaga dołączania do aplikacji dodatkowych bibliotek. Użytkownicy tego narzędzia mogą kontrolować sposób generowania klas poprzez adnotacje dodawane do XML Schema lub odrębny plik (`binding.xml`). Domyślnie, JAXB nie implementuje interfejsu *serializable*, można to jednak uzyskać poprzez dodanie odpowiednich adnotacji do XML Schema.

Narzędzie XML Beans jest częścią większego projektu Apache i jest dedykowane

wyłącznie do zastosowań opartych na XML Schema. Na uwagę zasługuje fakt, iż implementuje ono w pełni odwracalność operacji wiązania poprzez wsparcie wszystkich konstrukcji XML Schema oraz domyślnie implementuje interfejs *serializable*.

Wreszcie narzędzie JiBX zostało opracowane indywidualnie przez Dennisa Sosnoskiego z myślą o zapewnieniu jak najlepszej wydajności. W odróżnieniu od JAXB i XML Beans, w JiBX możliwe jest bezpośrednie odwzorowanie dokumentu XML na istniejącą strukturę obiektów – co wprawdzie wymaga dodatkowego nakładu pracy przy dekompozycji dokumentu XML, jednak później w trakcie działania aplikacji zapewnia bardziej bezpośredni dostęp do danych i umożliwia łatwiejsze tworzenie i modyfikację struktur danych poprzez użycie konstruktorów bez potrzeby korzystania z mechanizmów *object-factory*, jak to ma miejsce wypadku JAXB i XML Beans.

W implementacji architektury MIND w wersji wykorzystującej platformę JADE zostało wykorzystane narzędzie XML Beans. Wybór tego narzędzia był związany z faktem, że protokoły komunikacyjne systemów agentowych opartych na języku Java, do których zalicza się platforma JADE, wymagają implementacji interfejsu *serializable*, który jest domyślnie dodawany do klas generowanych tym narzędziem. Poza tym narzędzie to w pełni odzwierciedla wszystkie elementy XML Schema, przez co nie było potrzeby dostosowania wygenerowanych klas do prawidłowego przetwarzania dokumentów XML architektury MIND.

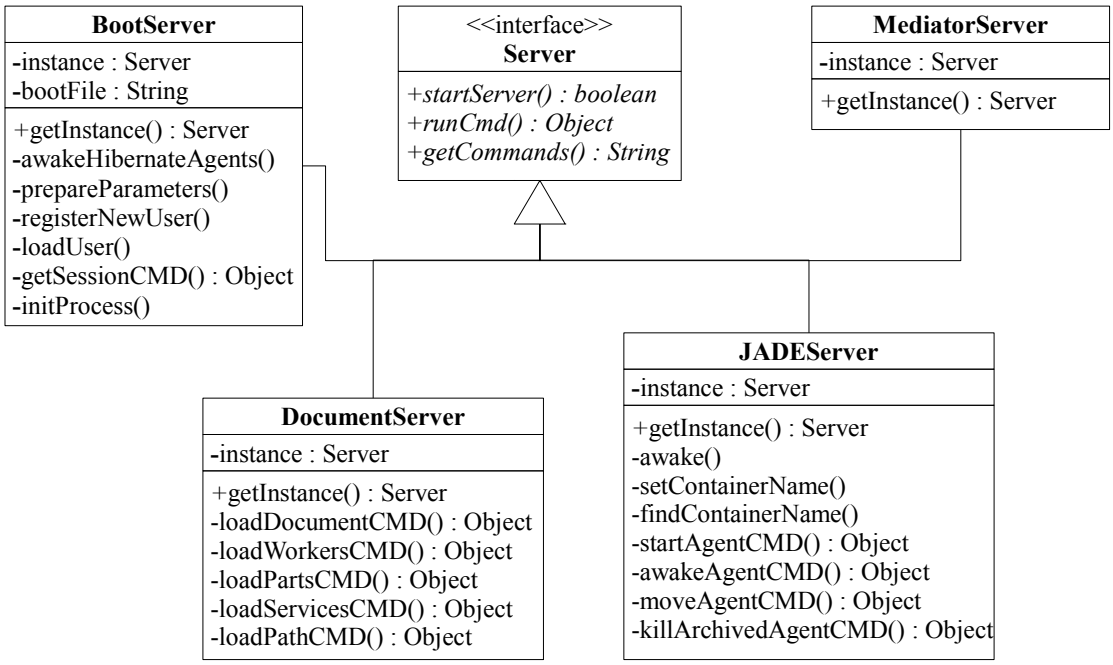
Do realizacji systemu klientów LWE zostało użyte narzędzie JAXB, gdyż wchodzi w skład standardowej platformy Java (nie trzeba dołączać dodatkowych bibliotek) i generuje mniejsze objętościowo zestawy klas, przez co bardziej pasuje do idei stworzenia „lekkiej" aplikacji klienckiej, nadającej się do użytku na urządzeniach mobilnych.

6.1.3. Realizacja architektury MIND w oparciu o system agentowy JADE

Model architektury MIND zapewnia możliwość zamiany statycznych dokumentów na zbiór dynamicznych obiektów (zob. rys. 4.1, str. 68). Omawiana w tym podrozdziale implementacja architektury MIND pozwala na przekształcenie dokumentów na postać obiektową i zarządzanie komponentami dokumentu rozproszonego w oparciu o platformę agentową JADE [Szc08]. Diagram głównych klas powstałej aplikacji przedstawia rysunek 6.3.

Aplikacja ta została napisana w języku Java. Inne technologie oparte na języku Java, wykorzystane w tej wersji prototypu obejmowały:

– system agentowy *JADE*, wykorzystany do tworzenia, migracji i śledzenia



Rys. 6.3. Klasy aplikacji prototypu MIND na platformie agentowej JADE

agentów,

– narzędzie *XML Beans* [Apa04], użyte do automatycznego wiązania danych dokumentów XML z obiektami Java,

– dodatkowe narzędzie *Log4j* [Apa12], wykorzystane do tworzenia dzienników zdarzeń aplikacji.

Klasy na rysunku 6.3 reprezentują podstawowe moduły systemu. Zostały one zrealizowane jako wewnętrzne serwery implementujące wspólny interfejs i udostępniające zestaw usług do wewnętrznego użytku systemu.

Moduł serwera platformy JADE

Klasa `JADEServer` jest głównym elementem modułu odpowiedzialnego za zarządzanie platformą JADE. Obiekt tej klasy zajmuje się uruchomieniem kontenera – środowiska agentowego JADE. Platformę JADE tworzy zbiór kontenerów, z których jeden pełni rolę *kontenera głównego*, uruchamiającego platformę (zob. rys. 6.2(a)).

Współpraca różnych stanowisk komputerowych w jednym systemie agentowym wymaga, by do kontenera głównego rejestrowały się pozostałe kontenery. Próba połączenia się z głównym kontenerem wymaga podania jego adresu IP. Pomyślne nawiązanie połączenia z głównym kontenerem decyduje o tym, czy system będzie

pracował w trybie *online* i miał możliwość przesyłania agentów, czy w trybie *off-line*, dającym możliwość jedynie lokalnej pracy z dokumentem.

Po uruchomieniu kontenera JADE, moduł serwera agentów jest odpowiedzialny za zarządzanie agentami. Na początku odpowiada za utworzenie agenta (`startAgentCMD`) lub „obudzenie", czyli odtworzenie z postaci zserializowanej na dysku (`awakeAgentCMD`). Tworzenie agenta odbywa się na podstawie pakietu MIND. Agent otrzymuje dane niezbędne do jego zbudowania, czyli unikatową nazwę oraz obiektową reprezentację pakietu MIND.

Przesłanie agentów do innego kontenera jest możliwie przy trybie pracy *online* względem platformy JADE. W komponencie pracowników dokumentu MIND umieszczone są nazwy ich kontenerów, zatem na podstawie danych zawartych w dokumencie wyznaczany jest docelowy kontener (`findContainerName`), na który ma być przesłany agent, zgodnie z jego polityką migracji. Moduł serwera JADE posiada metodę (`moveAgentCMD`) wykorzystującą mechanizm komunikacyjny platformy, która umożliwia przesłanie konkretnego agenta na wyznaczony kontener.

Moduł dokumentu MIND

Moduł dokumentu pozwala na przekształcenie składników pakietu MIND na postać obiektową i na realizację funkcjonalności zapisanej w dokumencie. Na podstawie pakietu MIND, przy pomocy narzędzie XML Beans zostały utworzone dwie biblioteki: `XMLHubDoc.jar` wygenerowana na podstawie schematu XML Schema definiującego format dokumentu MIND (zob. załącznik A) oraz `xpdl.jar` wygenerowana na podstawie schematu XML Schema dla formatu XPDL [WfM12a], który jest formatem zapisu komponentu przepływu pracy dokumentu MIND.

Biblioteki te pozwalają na wiązanie statycznych danych dokumentu z obiektami, zgodnie ze strukturą obiektową zamieszczoną na rysunku 4.1 (str. 68). Obiekt dokumentu staje się następnie polem agenta (obiektu klasy `MobileAgent`, która rozszerza klasę agenta JADE), dzięki czemu agent posiada wbudowane wszystkie dane sterujące określone w pakiecie MIND oraz binarną reprezentację odpowiedniego dokumentu składowego zawierającego treść informacyjną.

Moduł dokumentu udostępnia agentowi wszystkie niezbędne metody do zarządzania obiektami dokumentu, w tym za wydzielenie z obiektu dokumentu składowego i zapisanie go na lokalnym komputerze, tak by pracownik mógł wykonywać na tym dokumencie czynność, przypisaną mu w komponencie przepływu pracy. Po zakończonej pracy na dokumencie składowym, jego aktualna wersja zostaje ponownie wczytana w postaci binarnej przez agenta.

Zakończenie pracy z dokumentem na danym kontenerze powoduje automatyczne przeniesienie agenta na inny kontener. W tym celu moduł dokumentu dokonuje inter-

pretacji komponentu polityki migracji, przeszukuje tranzycje wyjściowe z aktualnie przetwarzanej czynności i wyznacza kontenery pracowników jako kolejne miejsca przeznaczenia.

Moduły dodatkowe

Dodatkowe moduły, przedstawione na diagramie klas (rys. 6.3) służą uruchamianiu i zarządzaniu systemu agentowego.

Moduł mediatora, w przeciwieństwie do innych serwerów, nie udostępnia żadnej operacji. Serwis ten zajmuje się jedynie zapewnieniem komunikacji pomiędzy poszczególnymi modułami i stanowi realizację wzorca projektowego o nazwie mediator [GHJV95], który zmniejsza liczbę powiązań między różnymi klasami, poprzez utworzenie jednej klasy (mediatora), znającej ich metody. Zatem poszczególne elementy systemu są zrealizowane jako serwery (klasy udostępniające zestaw metod). Nie muszą one nic o sobie wiedzieć – przekazują jedynie odpowiednie polecenia mediatorowi, a ten rozsyła je do odpowiednich obiektów innych klas.

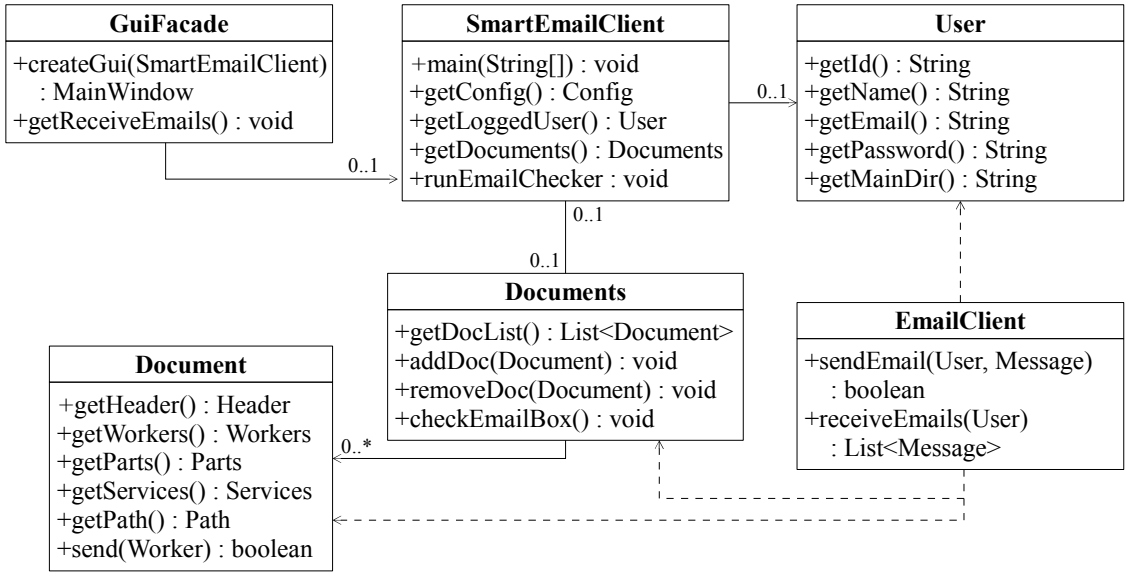
W celu uruchomiania poszczególnych podsystemów składowych, został wydzielony specjalny moduł inicjujący. Zapewnia on uruchomienie wszystkich podsystemów oraz ich inicjalizację (`initProcess`). Zadaniem modułu jest pobranie danych od użytkownika w trakcie logowania do systemu (`loadUser`). Poprawne wprowadzenie danych kończy się uruchomieniem systemu. Podczas pierwszego logowania (rejestracji użytkownika – `registerNewUser`) tworzony jest obiekt sesji użytkownika zapisywany w postaci pliku `Boot.ids`.

6.1.4. Realizacja architektury MIND w oparciu o klienty LWE

Klienty LWE zainstalowane jako lokalne aplikacje na urządzeniach uczestników procesu, tworzą luźno powiązany system rozproszony, w którym komunikacja odbywa się za pośrednictwem poczty elektronicznej, co zostało przedstawione na rysunku 6.2(b). Każdy klient LWE umożliwia agentom MIND ich dwa kluczowe zadania: komunikację treści i koordynację wykonywanych czynności. Komunikacja treści jest realizowana poprzez uruchamianie odpowiednich usług zadeklarowanych w dokumencie, które umożliwiają transformację zawartości dokumentów składowych. Koordynacja czynności odbywa się poprzez realizację pliku polityki migracji, będącego komponentem dokumentu MIND i na jego podstawie wykonywane jest przesyłanie pakietu MIND za pośrednictwem poczty elektronicznej do wyznaczonych pracowników.

Klient LWE (ang. *Local Workflow Engine*) został zaimplementowany zgodnie

z założeniami przedstawionymi w podrozdziale 5.2 i realizuje fazy cyklu życia dokumentu na pojedynczym urządzeniu pracownika zgodnie z diagramem stanów przedstawionym na rysunku 5.2 (str. 101). Rysunek 6.4 przedstawia uogólniony diagram klas klienta LWE [Czy13], które reprezentują jego główne moduły.



Rys. 6.4. Uogólniony diagram klas klienta LWE

Aplikacja klienta LWE została napisana w języku Java i wykorzystuje następujące technologie związane z tym językiem:

- platforma *Java Mail* [Ora13b] do utworzenia warstwy transportowej dla dokumentu MIND w oparciu o pocztę elektroniczną,
- narzędzie *JAXB* [Ora13c] do automatycznego wiązania danych dokumentów XML z obiektami Java,
- *Java Swing* wraz z komponentami *MigLayout* [MiG11] i *BaloonTip* [Ora13a] do utworzenia interfejsu użytkownika klienta LWE.

Klient LWE został zaimplementowany jako aplikacja wielowątkowa, w której osobne wątki odpowiadają za realizację interfejsu graficznego użytkownika, warstwę logiczną odpowiedzialną przede wszystkim za realizację funkcjonalności dokumentu MIND oraz komunikację z serwerami pocztowymi w celu realizacji mobilności dokumentu. Główna klasa klienta LWE – `SmartEmailClient`, pozwala na uruchomienie wszystkich modułów aplikacji.

Moduł poczty elektronicznej

Dokumenty są przekazywane między klientami LWE w postaci spakowanego pakietu MIND, stanowiącego załącznik wiadomości elektronicznej. Dopiero odebranie dokumentu przez klienta LWE, powoduje aktywowanie pakietu MIND, czyli przekształcenie go z postaci statycznej w obiektową. Moduł poczty elektronicznej (`EmailClient`) pośredniczy między klientem LWE a serwerami pocztowymi i odpowiada za odbieranie pakietów MIND z serwera pocztowego (`receiveEmails`) oraz wysyłanie na kolejny serwer pocztowy po zakończeniu jego misji na aktualnym urządzeniu pracownika (`sendEmail`).

Przypomnijmy, iż cykl życia dokumentu na pojedynczym urządzeniu (por. rys. 5.2) rozpoczyna się od odebrania dokumentu ze skrzynki odbiorczej i zapisaniu go na lokalnym urządzeniu pracownika. LWE posiada mechanizm uruchomiony w oddzielnym wątku aplikacji sprawdzający skrzynkę odbiorczą co pewien, ustalony w pliku konfiguracyjnym, odstęp czasu. Każdy użytkownik może ustalić dla swojego klienta LWE własny odstęp czasu (`runEmailChecker`), a także może sprawdzić dostarczenie dokumentu w dowolnej chwili poprzez interfejs graficzny klienta LWE.

W celu odebrania dokumentu ze skrzynki odbiorczej pracownika, klient LWE łączy się z serwerem poczty, przeszukuje listę wiadomości odebranych i tworzy listę tych, które są oznaczone jako nieprzeczytane. Z tej listy następnie wybiera tylko te wiadomości, które posiadają nagłówek o nazwie `X-SEC-MIND-ID` i wartości będącej identyfikatorem przesłanego dokumentu. Na tej podstawie tworzona jest lista odebranych wiadomości, które zawierają pakiet MIND.

Sam pakiet MIND, spakowany do formatu ZIP, znajduje się w załączniku. Odebrane wiadomości oznaczane są na serwerze jako przeczytane, zatem LWE nie będzie ich już pobierał przy kolejnym sprawdzaniu. Klient LWE nie usuwa wiadomości z serwera, na wypadek konieczności ich odtworzenia. Odbieranie wiadomości kończy się zapisaniem pliku załącznika, zawierającego pakiet MIND w katalogu `arrivals` na dysku użytkownika klienta LWE (pełna ścieżka do tego katalogu znajduje się w pliku konfiguracyjnym).

Dokument MIND, po wypełnieniu swojej misji na danym urządzeniu, jest wysyłany przez klienta LWE na adres pocztowy pracownika, który jest wykonawcą kolejnej czynności zgodnie z plikiem polityki migracji. Moduł poczty elektronicznej tworzy wiadomość elektroniczną w formacie MIME, która zawiera standardowe informacje o nadawcy, odbiorcy i temacie, a także nagłówek `X-SEC-MIND-ID` oraz spakowany pakiet MIND dodany jako załącznik.

Moduł dokumentu MIND

W pierwszej kolejności po odebraniu pakietu MIND z serwera pocztowego, moduł dokumentu rozpakowuje odebrany załącznik wiadomości i umieszcza w katalogu roboczym `working`. W tym katalogu może się znajdować wiele pakietów MIND, dotyczących różnych spraw. Klient LWE tworzy listę wszystkich dostarczonych dokumentów (`getDocList`). Poza tym nowo odebrane dokumenty również są na bieżąco dołączane do tej listy (`addDoc`).

Klient LWE posiada dwa pakiety klas: `dokument` i `xpdl`, które zostały wygenerowane przy pomocy narzędzia JAXB na podstawie schematów XML Schema opisujących architekturę MIND. Klasy te, analogicznie do rozwiązania z poprzedniej implementacji, umożliwiają modułowi dokumentu wiązanie dokumentów pakietu MIND z obiektami. Efektem tego jest możliwość utworzenia obiektu klasy `Document`, który zawiera wszystkie komponenty dokumentu MIND i może realizować swoją funkcjonalność, do której w szczególności należy realizacja polityki migracji i umożliwienie pracownikom lub określonym usługom transformacji treści dokumentów składowych.

Po wykonaniu wszystkich zadań przypisanych użytkownikowi danego LWE związanych z dokumentem, jest on serializowany do postaci pakietu MIND, a następnie pakowany do formatu ZIP w celu przesłania pocztą elektroniczną. Odbiorca wiadomości jest wyznaczany automatycznie na podstawie komponentu przepływu pracy.

Dodatkowo, moduł umożliwia zapisanie dokumentów po skończonym procesie na liście dokumentów archiwalnych oraz usuwanie dokumentów z lokalnego dysku pracownika, gdy nie są już potrzebne.

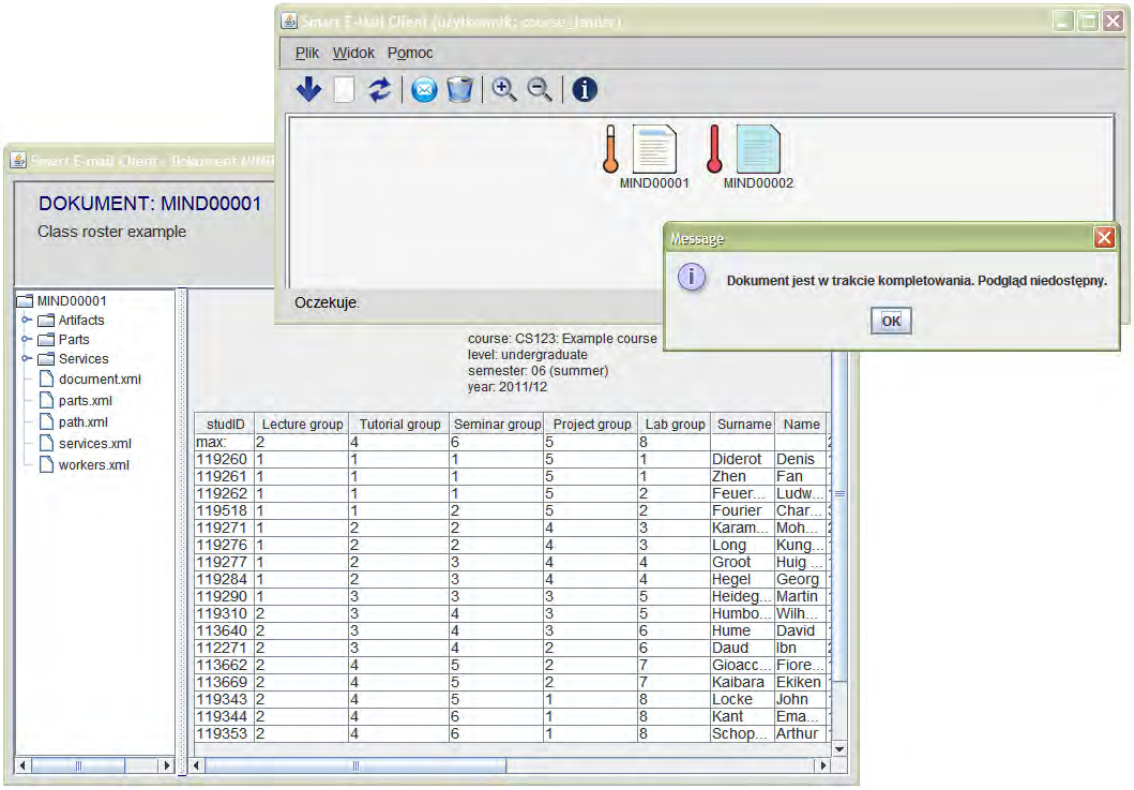
Moduły dodatkowe

Każdy klient LWE jest związany z konkretnym pracownikiem wiedzy, zalogowanym do aplikacji. Użytkownik może konfigurować niektóre ustawienia swojego klienta LWE, np. ustalić katalog roboczy dla dokumentów MIND czy częstotliwość automatycznego sprawdzania skrzynki odbiorczej.

Interfejs graficzny klienta LWE, obsługiwany przez klasę `GuiFacade`, służy natomiast do interakcji użytkownika z dokumentem MIND i klientem LWE. Jest on zaimplementowany w formie okna, w którym po uruchomieniu aplikacji, wyświetlane są dostępne dokumenty MIND. Dodatkowo, okno zawiera zestaw opcji, które pozwalają na uruchomienie sprawdzenia dostępności dokumentów w skrzynce odbiorczej oraz przejrzenie archiwum dokumentów.

Podwójne kliknięcie ikony dokumentu, powoduje wyświetlenie jego zawartości w oddzielnym oknie wraz ze strukturą jego katalogów. Interfejs użytkownika jest

przedstawiony na rysunku 6.5. Okno główne aplikacji w przedstawionym przykła-



Rys. 6.5. Graficzny interfejs klienta LWE

dzie zawiera dwa dokumenty: MIND00001 i MIND00002. Dokument MIND00001 został otworzony w oddzielnym oknie – sposób jego prezentacji został przyniesiony przez dokument jako usługa transformacji do postaci XHTML. Dokument MIND00002 nie mógł zostać otworzony, gdyż wykonywana jest na nim czynność łącząca dokumenty, a nie wpłynęły do klienta LWE jeszcze wszystkie części składowe potrzebne do scalenia dokumentu. Symbole termometrów oznaczają priorytety, które są związane z czasem pozostałym do wykonania czynności na danym dokumencie. Czas ten jest określony w pliku polityki migracji, w elemencie `<deadline>`, wczytywany i odliczany przez LWE w chwili rozpoczęcia wykonywania aktualnej czynności. Najniższy priorytet, oznaczany termometrem w kolorze zielonym, informuje użytkownika, że na wykonanie czynności ma więcej niż 7 dni. Wraz z upływem czasu, zmieniany jest kolor termometru: żółty oznacza normalny priorytet (3-7 dni), pomarańczowy oznacza wysoki priorytet (1-3 dni) i czerwony to priorytet krytyczny (mniej niż 1 dzień). Przedziały czasowe dla priorytetów mogą być, oczywiście, zmieniane w zależności od potrzeb organizacji.

6.1.5. Porównanie zrealizowanych prototypów

Omówiona wcześniej dwie różne realizacje prototypu systemu agentowego potwierdzają jednoznacznie realizowalność dwóch kluczowych cech architektury MIND – wykonywalności i mobilności.

Wykonywalność w obu rozwiązaniach wymagała wykorzystania operacji wiązania statycznego pakietu MIND z obiektami Java. Po tej operacji obiekty zachowują swoją autonomiczność poprzez automatyczną realizację komponentu przepływu pracy oraz zdolność do modyfikowania i uaktualniania swojej zawartości. Pozwalają też na transformację treści dokumentów składowych zarówno przez pracowników, jak i automatycznie przez dostępne w środowisku dokumentu usługi. Zastosowane narzędzia służące do automatycznego wiązania danych XML z obiektami (XML Beans i JAXB) sprawdziły się zarówno w przypadku deserializacji, jak i serializacji dokumentów XML.

Opisane rozwiązania znacząco różniły się w kwestii realizacji mobilności dokumentów MIND, choć oba pozwoliły na skuteczne przekazywanie dokumentów między rozproszonymi komputerami pracowników. Należy podkreślić, że realizacja architektury MIND w oparciu o platformę JADE miała jednak pewne wady. Pierwsza to zależność od zewnętrznej platformy, a zatem brak wpływu na jej rozwój. Platforma JADE rozwija się dość dynamicznie, a zmiany pojawiają się często, co ma znaczący wpływ na stabilność pierwszego z proponowanych przeze mnie rozwiązań. Ponadto, użytkownicy tej wersji prototypu muszą posiadać umiejętności zarządzania platformą JADE i rozwiązywania problemów związanych z jej działaniem oraz zdarzającymi się błędami [JAD13].

Jednym z problemów na jakie napotkałam w JADE był błąd serializacji. Mianowicie, czasem w trakcie wykonywania migracji agentów, platforma JADE zwracała błąd serializacji, w efekcie czego, u nadawcy agent był zabijany i nie docierał do adresata. Inny problem pojawiał się przy częstych zmianach stanu kontenera klienta, które powodowały problemy z migracją, tak jakby agent AMS nie odświeżał poprawnie stanu całej platformy. Pojawiał się także problem z poprawną obsługą żądań wysyłanych do zainstalowanego na lokalnym kontenerze agenta DF katalogującego usługi agentów. Przyjmował on rejestrację, ale nie odpowiadał poprawnie, zaś przy dużej licznie zapytań (rzędu 1000 agentów w systemie) potrafił stracić wydolność i zniknąć.

Rozwiązanie oparte na systemie klientów LWE wykorzystuje do realizacji mobilności agentów standardowe i powszechnie stosowane mechanizmy poczty elektronicznej. Niezależność od konkretnej platformy i wykorzystanie sprawdzonego mechanizmu komunikacyjnego determinuje trwałość tego rozwiązania. Poczta elektroniczna jest od dawna powszechnie używana do przesyłania dokumentów, a klienty LWE

wspierające architekturę dokumentu MIND skutecznie rozszerzają jej funkcjonalność w zakresie komunikacji treści i koordynacji procesów.

Ponadto, jeden system mogą tworzyć klienty LWE zaimplementowane w zupełnie różnych technologiach, także takich, które nie wykorzystują języka Java. Wystarczy, aby każdy klient potrafił realizować fazy cyklu życia dokumentu na pojedynczym urządzeniu pracownika wiedzy (zgodnie z diagramem zaprezentowanym na rys. 5.2) oraz umożliwiał wysyłanie i odbieranie dokumentów z serwerów pocztowych. Także przesyłanie dokumentu zserializowanego do postaci XML w skompresowanym pakiecie jest znacznie pewniejsze niż przesyłanie zserializowanego obiektu agenta. W przypadku awarii lokalnego środowiska agentowego, dokument w postaci XML pozostaje czytelny dla użytkownika i może być przetwarzany nawet w swojej statycznej postaci.

6.2. Realizacja procesów wiedzy w architekturze MIND

Procesy wiedzy są realizowane w oparciu o trzy zasoby wiedzy organizacji: wiedzę ugruntowaną, skodyfikowaną i spersonalizowaną (zob. podrozdział 2.2). Istotne jest zatem, aby agenty MIND potrafiły zrealizować strukturę każdego procesu przepływu pracy, uwzględniając interakcję z pracownikami i dając możliwość archiwizacji dokumentów w repozytoriach. Realizację struktury procesu wiedzy zapewnia implementowalność kanonicznego zbioru wzorców koordynacji dokumentów, przedstawionego w podrozdziale 5.3. Interakcja z pracownikami jest realizowana poprzez usługi wykonywane w ramach czynności procesu. Zaś możliwość archiwizacji dokumentów jest związana z trwałością formatu przechowywania danych i specyfikacji struktury logicznej dokumentu.

6.2.1. Specyfikacja szablonu struktury logicznej

Dokument MIND został zapisany w języku XML, który jest powszechnie używanym standardem jednolitej notacji do opisu treści dokumentu zgodnym z pewnym odpowiednio wyspecyfikowanym szablonem struktury logicznej, czytelnym dla człowieka i komputera. Specyfikacja struktury logicznej dokumentu ma dla architektury MIND znaczenie kluczowe, bowiem bezpośrednio na jej podstawie odbywa się dekompozycja dokumentu na obiekty składowe (komponenty) podlegające dalszemu przetwarzaniu podczas swej migracji w systemie.

Istnieje wiele notacji do specyfikacji struktury logicznej dokumentu XML o zróżnicowanej składni, a z punktu widzenia trwałości modelu przetwarzania opartego

na architekturze MIND, ważny jest wybór takiej notacji, która przetrwa jako standard i będzie wspierana przez dostawców oprogramowania i technologii bazujących na XML (lub daje możliwość transformacji do innych notacji), tak by dokumenty wytwarzane dzisiaj były kompatybilne z technologiami jakie zostaną wprowadzone w przyszłości. Ma to znaczenie dla zastosowań architektury MIND w takich dziedzinach jak sądownictwo, medycyna czy bezpieczeństwo transportu, w których repozytoria dokumentów stanowią podstawowe zasoby wiedzy niezbędnej do podejmowania decyzji w trybie obliczeń zespołowych oraz utrzymują swoją ważność przez wiele lat (np. akty notarialne). Poniżej scharakteryzuję trzy najpopularniejsze notacje do opisu schematu logicznego dokumentów XML: DTD, XML Schema i Relax NG.

Notacja *Document Type Definition (DTD)* [Wor98], wywodząca się z języka SGML (protoplasty XML), ma bardzo prostą składnię, pozwalającą definiować zagnieżdżone struktury z możliwością parametryzacji. Choć ta notacja jest wspierana przez większość dostawców oprogramowania i ma perspektywy przetrwania jako domyślny format specyfikacji struktury logicznej dokumentów XML, jej siła wyrazu nie jest duża. Składnia DTD z formalnego punktu widzenia należy do klasy języków LTG (ang. *local tree grammar*) stanowiących podklasę języków RTG (ang. *regular tree grammar*)[MLMK05]. W gramatykach LTG nie występuje konflikt symboli nieterminalnych, tzn. dla żadnej pary reguł produkcji nie występuje po prawej stronie ten sam symbol nieterminalny. W ogólności, występowanie konfliktu symboli nieterminalnych w gramatyce języka opisu szablonu utrudnia walidację tworzonych dokumentów XML, bowiem możliwa jest dla takiej gramatyki więcej niż jedna prawidłowa interpretacja dokumentu.

Intencją twórców notacji *XML Schema* [Wor12] wspieranej od 2001 przez organizację W3C było wprowadzenie do definicji szablonu mechanizmów umożliwiających definiowanie parametrów i elementów struktury logicznej dokumentu jako typów, z możliwością ich dziedziczenia, rozszerzania i zawężania. Gramatyka XML Schema jest wprawdzie mało intuicyjna i ogólnie trudniejsza w użyciu, ale za to mniej restrykcyjna od DTD, gdyż należy do ogólniejszej od LTG klasy STG (ang. *single-type tree grammar*), tzn. LTG $\subsetneq$ STG $\subsetneq$ RTG. W gramatykach STG konflikt symboli nieterminalnych może występować co najwyżej tylko dla par reguł produkcji dotyczących jednego modelu zawartości. Schematy logiczne wielu dialektów XML (w tym XPDL) są opisane właśnie w XML Schema.

Najmniej restrykcyjnym językiem opisu szablonu struktury logicznej dokumentu jest *Relax NG* [Org01], wspierany przez organizację OASIS. Gramatyka Relax NG formalnie należy do klasy RTG, w której prawa strona reguły produkcji może być dowolnym wyrażeniem regularnym symboli nieterminalnych. Ta „dowolność" utrud-

nia wykrywanie niejednoznaczności przy dopasowywaniu wyrażeń podczas analizy zgodności dokumentu z szablonem. Pierwsze próby prostego rozszerzenia znanych algorytmów walidacji dokumentów względem szablonów opisanych językami z klas LTG czy STF do klasy RTG zakończyły się fiaskiem, prowadząc do algorytmów niedeterministycznych, niezdolnych do zapewnienia unikatowej interpretacji dokumentu. Jednak później zaproponowano algorytmy walidacji dokumentów dla szablonów opisanych dowolnym językiem klasy RTG, które wykorzystują „zachłanną" metodę dopasowywania wyrażeń regularnych [FC04], co doprowadziło do zwiększenia zainteresowania językiem Relax NG i pozwoliło przyśpieszyć rozwój narzędzi do przetwarzania dokumentów XML.

Relax NG występuje również wersji *Relax NG Compact System (Relax NC)* [Org02], nie będącej dokumentem XML, która jest oparta na notacji BNF. Zaletą takiej notacji jest mniejsza „nadmiarowość" danych opisujących strukturę logiczną i lepsza czytelność dla użytkowników – a więc eliminuje główne zarzuty oponentów języka XML. Podobnie, na notacji BNF jest oparty format dokumentów *JavaScript Object Notation (JSON)* [Cro06], zyskujący coraz większą popularność względem formatu XML i przez mniejszą objętość pliku, znajdujący zastosowanie szczególnie w aplikacjach na urządzenia mobilne. Jest to jednak język bazujący na strukturze XML i umożliwiający automatyczną dwustronną konwersję do formatu XML. Do specyfikacji struktury dokumentów JSON może być używana notacja *JSON Schema* [Zyp11] – bazująca na XML Schema i Relax NG.

Odnośnie trwałości architektury MIND istotne jest istnienie algorytmów umożliwiających transformację między różnymi formatami opisu struktury logicznej dokumentu. Przykładami tych możliwości jest algorytm automatycznej konwersji szablonu dokumentu z formatu XML Schema na format Relax NG, zaimplementowany i dostępny publicznie w postaci arkusza XSL [Deb11] oraz narzędzie *Trang* [Tha08] zaimplementowane w języku Java, służące do konwersji między notacjami Relax NG, Relax NC, DTD i XML Schema (z tym, że XML Schema może być tylko formatem wyjściowym).

Do opisu struktury logicznej dokumentu MIND została użyta notacja XML Schema ze względu na możliwość wykorzystania jej przez narzędzia do automatycznego wiązania danych z obiektami, które zostały wybrane do realizacji prototypów systemu opartego na architekturze MIND.

6.2.2. Realizacja wzorców koordynacji dokumentu

Komponenty dokumentu MIND przenoszą jego części składowe w trakcie migracji między stacjami roboczymi poszczególnych pracowników wiedzy w celu umożli-

wienia im wykonania określonych czynności wyspecyfikowanych w ścieżce migracji. Ten proces można formalnie opisać za pomocą języka przepływu pracy, wiążącego czynności wykonywane na poszczególnych stacjach z przejściami między nimi. Powiązania czynności i tranzycji tworzą zbiór powtarzających się wzorców przepływu sterowania, przedstawionych w podrozdziale 3.2.2, które pozwalają na porównywanie między sobą języków i technologii przepływu pracy pod kątem ich zdolności do opisu dowolnych scenariuszy migracji [ARH11].

Jak już wspominałam we wcześniejszych rozdziałach, do implementacji architektury MIND zastosowałam język XPDL wspierany przez organizację WfMC [WfM12b]. Jego ważną zaletą z punktu widzenia trwałości rozwiązań zaproponowanych przez MIND jest składnia oparta na XML, pozwalająca na zastosowanie transformacji XSLT do przekształcenia specyfikacji ścieżki migracji na inny język opisu przepływu pracy – co będzie ważne w przyszłości, gdy pojawią się jakieś nowe standardy opisu przepływów pracy.

Z punktu widzenia realizacji architektury MIND, istotna w wyborze języka przepływu pracy była też możliwość implementacji kanonicznego zbioru wzorców koordynacji dokumentu, opisanego w podrozdziale 5.3. W XPDL, poprzez elementy `ExtendedAttribute`, można dodawać zmienne sterujące, pozwalające na realizację wymienionych wzorców w przypadku rozproszonego zarządzania przepływem pracy (zob. podrozdział 5.1). Ponadto, plik przepływu pracy jest również dokumentem zamienianym na komponent każdego agenta MIND, co umożliwia dynamiczną modyfikację jego ścieżki migracji przez uprawnionego pracownika wiedzy, w trakcie wykonywania danej czynności. Następnie, wszystkie zmiany można obejrzeć w statycznej wersji dokumentu XPDL i dzięki temu prześledzić oraz zarchiwizować przebieg procesu.

Między innymi dlatego zastosowanie XPDL wydaje się być lepszym rozwiązaniem, niż bezpośrednie wykorzystanie języka programowania do implementacji przepływów pracy, co umożliwia np. platforma WADE (*Workflow and Agents Development Environment*) [Tel13], stanowiąca rozszerzenie platformy JADE. Agent JADE ma wbudowany dodatkowy obiekt zdolny do wykonywania czynności i przejść opisanych formalnie (zaprogramowanych) w języku Java. Rozwiązanie zaproponowane przez twórców WADE pozwoliłoby zwiększyć funkcjonalność obiektów MIND, jednak utrudniłoby dynamiczną modyfikację opisu ścieżki migracji w trakcie realizowania procesu.

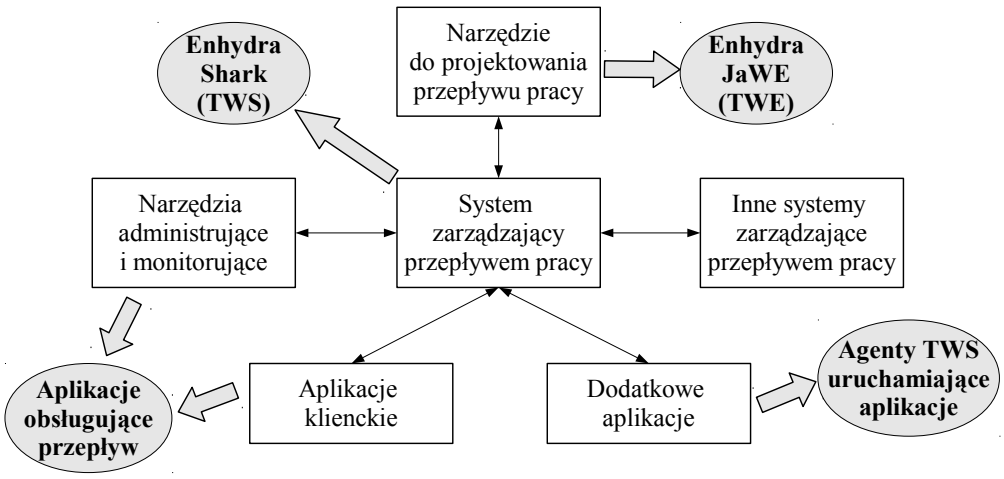
Wzorce koordynacji dokumentów zostały opracowane na podstawie kanonicznego zbioru wzorców przepływu sterowania [ARH11] (podrozdział 3.2.2) i w pierwszej kolejności zostały przetestowane na lokalnym środowisku z wykorzystaniem ogólnodostępnej, centralnej usługi zarządzającej przepływem pracy. Miało to na celu szybkie

wykonanie pierwszego prototypu (ang. *rapid prototyping*), pozwalającego na weryfikację opracowanych wzorców w odniesieniu do procesu przepływu dokumentów oraz ustalenie, jaką funkcjonalność powinna mieć lokalna aplikacja realizująca przepływ pracy, wchodząca w skład rozproszonej usługi zarządzającej przepływem pracy dokumentu, przedstawionej na rysunku 5.1(b).

Realizacja przepływu pracy dokumentu na jednym urządzeniu, redukuje fazy cyklu życia dokumentu na lokalnym środowisku użytkownika (zob. rys. 5.2) o operacje związane odbieraniem i wysyłaniem pakietu MIND. Centralna usługa jest też uruchamiana raz dla przeprowadzenia całego procesu, zatem wyznaczanie aktualnych i kolejnych czynności jest wykonywane na podstawie lokalnych zmiennych usługi i przez to do pliku polityki migracji nie muszą być dopisywane dodatkowe atrybuty sterujące przepływem pracy (`CURRENT_ACTIVITY` i `MAIN_PROCESS`), konieczne przy rozproszonym zarządzaniu procesem.

Usługa zarządzająca przepływem pracy

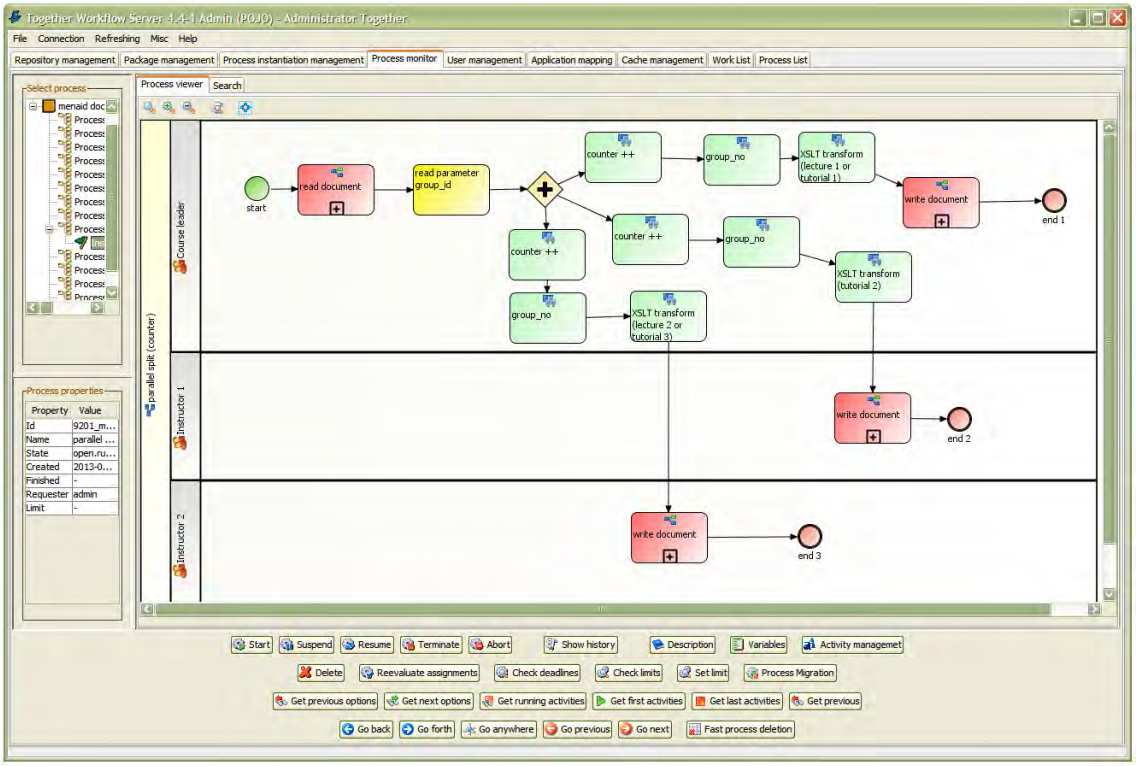
Do realizacji pierwszego prototypu wzorców koordynacji dokumentu MIND, wybrałam system zarządzający przepływem pracy *Together Workflow Server (TWS)*, znany też pod nazwą *Enhydra Shark* [Tog11a], gdyż pozwala on na automatyzację procesów zaimplementowanych bezpośrednio w języku XPDL. W pakiecie narzędziowym *Together* znajduje się także edytor XPDL i BPMN – *Together Workflow Editor (TWE)*, nazywany też *Enhydra JaWE* [Tog11b], który pozwala tworzyć diagramy BPMN, zapisywać je bezpośrednio w formacie XPDL i uruchamiać w Enhydra Shark. Komponenty TWS i TWE są przedstawione na rysunku 6.6 i realizują one w całości model referencyjny systemu zarządzającego przepływem pracy zaproponowany przez Workflow Management Coalition (WfMC) [WfM12b].



Rys. 6.6. Komponenty TWS i TWE

System TWS udostępnia kilka rodzajów aplikacji do uruchamiania, administrowania i monitorowania przepływu pracy: *Swing Admin Application* to aplikacja okienkowa zrealizowana w Java Swing, *Web Client Application* to aplikacja instalowana na serwerze, którą można uruchamiać przez przeglądarkę internetową, *JSP Client Application* to bardzo prosta graficzna aplikacja zaimplementowana jako pojedyncza strona JSP, a *Console Client Application* to aplikacja nie posiadająca interfejsu graficznego, uruchamiana i obsługiwana z linii poleceń konsoli systemowej. Można również zintegrować „silnik" TWS z klientem poczty Microsoft Outlook.

Do testowania działania wzorców koordynacji dokumentów wykorzystałam aplikację *Swing Admin Application*, która posiada przejrzysty interfejs graficzny i udostępnia wiele narzędzi do administrowania i śledzenia procesu przepływu pracy na lokalnym środowisku. Rysunek 6.7 prezentuje interfejs wybranej aplikacji, a dokładniej zakładkę pozwalającą na monitorowanie uruchomionego procesu.



Rys. 6.7. Interfejs aplikacji TWS do zarządzania procesem

TWS posiada interfejs do uruchamiania dodatkowych aplikacji zaimplementowany w formie agentów pośredniczących między różnymi narzędziami. Mechanizmy te wykorzystałam do dostosowania „silnika" przepływu pracy do obsługi przepływu dokumentów.

Mechanizm *JavaClass Tool Agent* pozwala na implementację i wykorzystywanie własnych klas Java. Dodałam zatem klasy odczytujące, zapisujące i usuwające

dokument z dysku, a także klasę łączącą dokumenty w jeden, co stanowi pierwszy krok do scalenia rozłączonych dokumentów. Mechanizm *XSLT Tool Agent* pozwala na uruchamianie transformacji XSLT, co pozwoliło na wykonanie transformacji wbudowanych w dokument MIND, które rozdzielają dokument na części oraz scalają wcześniej rozdzielone części. Mechanizm *JavaScript Tool Agent* pozwala na uruchamianie prostych poleceń JavaScript, umieszczonych bezpośrednio w pliku XPDL. Wykorzystałam ten mechanizm do prostych operacji przypisywania wartości zmiennym zdefiniowanym w procesie. Są dostępne też agenty do uruchamiania serwisów sieciowych oraz wysyłania i odbierania poczty elektronicznej.

System TWS realizuje proces, rozpoczynając go od czynności startowych i kończąc w chwili osiągnięcia przez wszystkie aktywne wątki czynności końcowych lub po wymuszeniu przerwania przez osobę nadzorującą proces. Proces jest zarządzany centralnie i realizowany na jednym komputerze, zatem brak jest czynności wysyłających i odbierających dokumenty z serwera poczty elektronicznej. Przetwarzanie dokumentu rozpoczyna czynność read document, która wczytuje dokument składowy zapisany na dysku i przypisuje go jako wartość zmiennej `fileXMLcontent`. Czynnością odwrotną jest write document, która zapisuje wartość zmiennej `fileXMLcontent` do pliku XML i umieszcza go w strukturze katalogów stworzonej dla wskazanego w procesie pracownika wiedzy. Obie czynności są zdefiniowane jako podprocesy, które oprócz operacji na plikach, kontrolnie wyświetlają utworzone zmienne.

Wszystkie zmienne zadeklarowane w pliku przepływu pracy, są wczytywane jako zmienne lokalne „silnika" i nie ma możliwości edycji tego pliku w trakcie trwania procesu. Wzorce projektowane były jednak z myślą o rozproszonym zarządzaniu przepływem pracy, co pozwoliło na identyfikację zmiennych, które powinny być wpisywane do pliku przepływu pracy i wraz z nim przenoszone na kolejne lokalizacje. Następnie zasadność tych zmiennych została zweryfikowana w rozproszonym systemie realizującym architekturę MIND.

W tej części rozprawy przedstawię realizację wzorców koordynacji dokumentów, które zostały zaprojektowane i sprawdzone w narzędziach TWE i TWS i zweryfikowane w środowisku rozproszonym. Natomiast wszystkie diagramy BPMN utworzone w TWE i działające w systemie Enhydra Shark (TWS) wraz z opisami umieściłam w załączniku B.

Realizacja wzorców z rozproszonym stanem dokumentu

Do realizacji wzorców z rozproszonym stanem dokumentu wystarczą informacje przyniesione przez dokument i funkcjonalność lokalnego środowiska agentowego pozwalająca na właściwą interpretację danego wzorca.

Wzorzec sekwencyjny opisuje sytuację przekazania dokumentu innemu pracownikowi. Dokument ten może być przekazany za jednym razem w całości lub w częściach. W systemie TWS do realizacji tego wzorca zostały utworzone dwie zmienne procesu: `counter` i `continue`. Początkowa wartość zmiennej `counter` wynosi zero i jest ona zwiększana o jeden wraz wydzieleniem każdej części dokumentu. Jeśli dokument jest przesyłany w całości, wartość zmiennej `counter = 1`. Zmienna `continue` ma wartość początkową `true` i dopóki nie zostanie zmieniona na `false`, będą tworzone kolejne części dokumentu. Części dokumentu są przekazywane kolejnemu pracownikowi od razu po wydzieleniu, zaś ich scalenie następuje po zmianie wartości zmiennej `continue` na `false`.

W przypadku rozproszonego zarządzania przepływem pracy, kolejne wartości zmiennej lokalnej `counter` są wpisywane jako atrybut do pliku przepływu pracy każdej części dokumentu, zaś zmiana wartości zmiennej `continue` na `false` powoduje wpisanie do pliku przepływu pracy ostatniej części dokumentu atrybutu znacznika końca. W przypadku przesyłania całości dokumentu, gdy wartość zmiennej `counter = 1`, nie ma konieczności wprowadzania tych atrybutów do pliku polityki.

Poniższy fragment kodu prezentuje atrybuty dodane do pliku przepływu pracy ostatniej, 5. części przekazywanego dokumentu.

```
<ExtendedAttributes>
   <ExtendedAttribute Name="DOC_PART_NUMBER" Value="5"/>
   <ExtendedAttribute Name="SENTINEL" Value="true"/>
</ExtendedAttributes>
```

Części dokumentów mogą być dostarczane do punktu łączenia w różnej kolejności, stąd sam znacznik końca nie wystarcza do określenia, czy wszystkie dokumenty już wpłynęły.

Wzorzec rozdzielający dokument zrealizowany w TWS posiada czynność, która pobiera od użytkownika wartość parametru, według której będzie dokonywana transformacja rozdzielająca dokument na części. Interakcja z użytkownikiem jest możliwa, dzięki dodaniu do czynności dodatkowego atrybutu: `VariableToProcess_UPDATE`, zdefiniowanego w TWS do przypisywania zmiennym wartości podanych przez użytkownika i odbywa się poprzez okno dialogowe. W testowanym przykładzie, realizującym studium przypadku opisane w podrozdziale 6.3, użytkownik mógł wpisać lecture (wykład) lub tutorial (ćwiczenia). Wpisanie lecture spowodowało wybranie dwóch z trzech możliwych tranzycji, które miały warunek: `(group_id == "lecture") || (group_id == "tutorial" )`. Dokument trafił więc do dwóch wyznaczonych pracowników. Jednak zanim dokument został

przekazany odpowiednim pracownikom, dla każdej ścieżki zostały wykonane następujące czynności:

- `counter++` to czynność uruchamiająca aplikację dodającą dwie liczby; zwiększa ona wartość zmiennej `counter` o jeden, dla każdej wybranej ścieżki,
- `group_no` to czynność przypisująca wartość zmiennej, która będzie parametrem transformacji dokumentu;
- `XSLT transform` uruchamia transformację rozdzielającą dokument i zgodnie z ustalonymi parametrami, dokonuje podziału wczytanego dokumentu na części.

Wartość zmiennej `counter` jest wyznaczana dynamicznie, na podstawie wartości zmiennej, którą podał pracownik. W punkcie łączenia rozdzielonych dokumentów ta informacja będzie niezbędna i w przypadku rozproszonego zarządzania przepływem pracy musi zostać wpisana do ścieżki migracji każdej części dokumentu. Wartość atrybutu w omówionym przykładzie wynosi 2:

```
<ExtendedAttributes>
  <ExtendedAttribute Name="COUNTER" Value="2"/>
</ExtendedAttributes>
```

Wzorzec łączący dokument jest komplementarny z wzorcem rozdzielającym i na początku jego realizacji potrzebne jest ustalenie, ile dokumentów ma zostać scalonych. W TWS wartość ta jest wczytywana do zmiennej na samym początku jako formalny parametr o nazwie `counter`. Wczytywany jest też parametr, względem którego był rozdzielany dokument w celu wybrania odpowiedniej ścieżki. Wpłynięcie każdego dokumentu zmniejsza wartość zmiennej `counter` o jeden. Gdy osiągnie ona wartość zero, oznacza to, że wszystkie oczekiwane dokumenty wpłynęły i mogą zostać scalone.

W rozproszonym zarządzaniu przepływem pracy, licznik aplikacji lokalnej jest ustalany na podstawie atrybutu `COUNTER` pierwszego z odebranych dokumentów. Czynność łącząca pozostaje aktywna. Kolejne dokumenty powodują zmniejszanie licznika o jeden aż do zera. Wówczas czynność łącząca zostaje zakończona a dokument scalony.

W przypadku realizacji wzorca dyskryminującego łączenia, pojawia się dodatkowy atrybut `COUNTER2`, ustalający licznik dokumentów, które mają zostać zaakceptowane.

Wzorzec iteratora został zrealizowany w TWS poprzez dodanie zmiennej `continue`, której wartość jest ustalana przez użytkownika w punkcie decyzji o kontynuowaniu pętli.

W przypadku systemu rozproszonego mogą wystąpić dwa przypadki. Decyzja o kontynuacji pętli może być podjęta w trakcie wykonywania czynności, po której następuje interpretacja warunków przebiegu pętli. Wówczas, na podstawie tej decyzji, wyznaczana jest kolejna czynność i dokument zostaje przesłany do przypisanego jej pracownika. Ta sytuacja nie wymaga dodawania atrybutów do pliku przepływu pracy. Jeśli zaś decyzja o przebiegu pętli jest podejmowana w czasie realizacji wcześniejszej czynności, musi być zapisana w pliku polityki migracji jako atrybut:

```
<ExtendedAttributes>
   <ExtendedAttribute Name="LOOP_CONTINUE" Value="true"/>
</ExtendedAttributes>
```

Realizacja wzorców ze sprzężonym stanem dokumentu

Aplikacje TWS i TWE realizują tylko pewien podzbiór elementów XPDL, zbliżony do wersji XPDL 1.0. Między innymi nie obsługują czynności wysyłających i odbierających sygnały. Stąd do realizacji wzorców ze sprzężonym stanem dokumentu postanowiłam wykorzystać obsługiwany w TWS element `<Deadline>`, którego nominalnym przeznaczeniem jest przerwanie czynności po pewnym, zadeklarowanym w tym elemencie okresie czasu.

Wzorzec odroczonego wyboru wykorzystuje czynność `wait for active KW`, oczekującą na decyzję o wyborze pracownika, który pierwszy pojawi się jako uczestnik procesu, czyli wykona czynność `set active`, zmieniająca wartość zmiennej procesu `activeKW` na identyfikator tego pracownika. Zmienna `activeKW` może być zmieniona tylko w przypadku, gdy jej wartość wynosi zero, a więc gdy nie zmienił jej wcześniej inny pracownik. Pozwala to na wybranie pierwszego zgłoszonego pracownika, nawet jeśli obaj zgłosili się przed uruchomieniem czynności `wait for active KW`.

Czynność `wait for active KW` ma zdefiniowany element `<Deadline>`:

```
<xpdl:Deadline Execution="SYNCHR">
 <xpdl:DeadlineDuration>
   var d=new java.util.Date();
   if (activeKW != 0)
     d.setTime(ACTIVITY_ACTIVATED_TIME.getTime());
   d;
```

```
    </xpdl:DeadlineDuration>
    <xpdl:ExceptionName>choice_done</xpdl:ExceptionName>
  </xpdl:Deadline>
```

TWS daje możliwość automatycznego sprawdzania tego elementu. Zatem zmiana wartości zmiennej `activeKW` na wartość różną od zera, powoduje przerwanie wykonywania czynności `wait for active KW`, poprzez ustawienie czasu jej wykonywania (`ACTIVITY_ACTIVATED_TIME`) na 0 sekund. W wyniku przerwania, zostaje wykonana tranzycja wyjątku, a następnie, zgodnie z wartością zmiennej `activeKW`, dokument trafia do wybranego pracownika.

W przypadku rozproszonego zarządzania przepływem pracy nie istnieje zmienna globalna `activeKW`, do której mieliby dostęp wszyscy uczestnicy procesu. Również zastosowanie elementu `<Deadline>` do odbierania sygnałów nie jest zgodne z przeznaczeniem tego elementu opisanym w specyfikacji WfMC [WfM12b]. Zamiast tego można zastosować atrybut, który po wczytaniu przez lokalny „silnik" przepływu pracy, pełni rolę binarnego semafora. W chwili odebrania pierwszego sygnału, semafor zezwala na dalszy przepływ dokumentu. Początkowa wartość atrybutu semafora wynosi `true`, co oznacza zablokowanie wykonywania czynności. Atrybut ten jest przypisany do czynności, a nie do procesu, tak jak to było w przypadku poprzednich atrybutów.

```
<ExtendedAttributes>
   <ExtendedAttribute Name="LOCK" Value="true"/>
</ExtendedAttributes>
```

W najprostszej wersji implementacji, sygnał jest odbierany przez pracownika wiedzy, np. w formie SMS, rozmowy telefonicznej lub wiadomości elektronicznej przesłanej niezależnie od pakietu MIND i to pracownik wprowadza odpowiednią wartość zmiennej, na podstawie której podejmowana jest decyzja o dalszym przebiegu procesu. Automatyczne wysyłanie sygnałów przez lokalne środowiska może odbywać się również za pośrednictwem poczty elektronicznej w formie krótkich wiadomości zawierających informację o dokumencie i czynności, której dotyczą oraz podjętej decyzji.

Wzorzec kamienia milowego, jeśli chodzi o charakter czynności, jest podobny do wzorca odroczonego wyboru. Również wykorzystywany jest semafor, natomiast wysyłany sygnał nie musi zawierać w sobie żadnej informacji, gdyż jego zadaniem jest jedynie odblokowanie czynności.

Wzorzec anulujący w TWS został zrealizowany również w wykorzystaniem elementu `<Deadline>`, jako rozwiązania pozwalającego na odebranie „sygnału anulującego", czyli pozwalającego na przerwanie wykonywania czynności w sytuacji zmiany wartości zmiennej `cancel` z `false` na `true`. Anulowanie czynności powoduje przejście do czynności usuwającej dokument.

W systemie rozproszonym nie ma potrzeby stosowania semafora, gdyż czynność na początku nie musi być blokowana. Anulowanie czynności przerywa jej wykonywanie na lokalnym środowisku i usuwa dokument, co jest najprostszym rozwiązaniem, ale nie zawsze właściwym. Problemy związane z anulowaniem czynności zostały już omówione w podrozdziale 5.3.2.

Realizacja podprocesów

TWS potrafi realizować dwa rodzaje podprocesów:

- podprocesy globalne, zadeklarowane w odrębnym elemencie `<WorkflowProcess>` (zob. rys. 4.10, str. 87) i zagnieżdżone w czynnościach typu `SubFlow`, które mogą być wykorzystywane wielokrotnie w różnych procesach, a także jako samodzielne procesy;
- podprocesy lokalne, zdefiniowane jako zbiór czynności (`<ActivitySet>`) i zagnieżdżone w czynnościach typu `BlockActivity`, które mogą być wykorzystywane tylko w obrębie danego procesu i nie stanowią odrębnego procesu.

Jednak brak możliwości edycji procesu w trakcie jego trwania, nie pozwala na dodawanie nowych podprocesów zdefiniowanych przez uczestników procesu.

Rozproszone zarządzanie przepływem pracy daje możliwość definiowania nowych podprocesów i zagnieżdżania ich w obrębie jednej czynności w trakcie wykonywania procesu.

6.2.3. Wnioski z weryfikacji wzorców

Aplikacje TWS i TWE pozwoliły na zaprojektowanie i sprawdzenie wzorców przepływu sterowania [ARH11] (zob. podrozdział 3.2.2) pod kątem przepływu dokumentów. Dzięki temu został zaprojektowany i zweryfikowany kanoniczny zbiór wzorców koordynacji dokumentu, omówiony w podrozdziale 5.3, który pozwala na skonstruowanie dowolnego procesu wiedzy.

Wzorce te zostały następnie zrealizowane w oparciu o rozproszoną usługę zarządzającą przepływem pracy. Pozwoliło to na rozstrzygnięcie, jaką funkcjonalność musi zapewniać lokalna usługa zarządzająca dokumentami oraz jakie dane powinien

przenosić sam dokument, aby realizacja jego polityki migracji była możliwa do wykonania.

Lokalna usługa zarządzająca przepływem pracy musi zatem:

- zapewniać możliwość zmieniania wartości atrybutów komponentu przepływu pracy,
- dostosowywać swoje działanie do wartości dostarczonych atrybutów (realizacja licznika, semafora, oczekiwanie na dokumenty w punkcie łączenia),
- uruchamiać usługi mające na celu transformację treści dokumentu wykonywaną automatycznie lub w interakcji z pracownikiem,
- dawać możliwość dodawania i wykonywania podprocesów,
- pozwalać na odbieranie i wysyłanie sygnałów.

Wnioski te pozwoliły całościowo ocenić poprawność implementacji obu wcześniej opisanych prototypów.

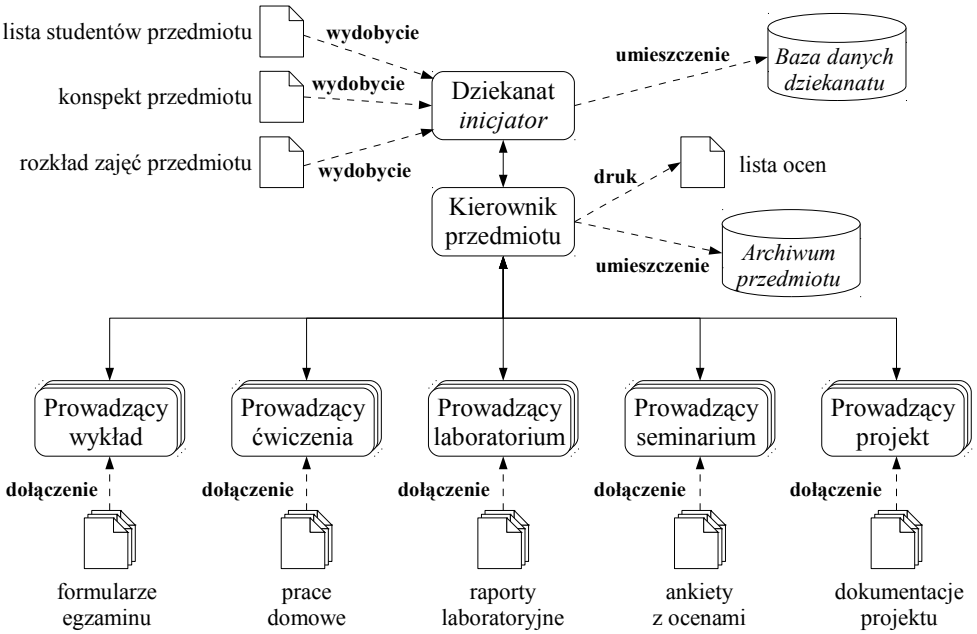
6.3. Studium przypadku – zarządzanie przedmiotem szkoły wyższej

Architektura MIND może być wykorzystywana we wszystkich organizacjach, w których występuje wymiana informacji między pracownikami poprzez dokumenty i pracownicy na podstawie swojej wiedzy i dostarczonych informacji zmieniają ich treść. Poprzez taką wymianę rozwiązywane są złożone, niealgorytmiczne problemy decyzyjne, których istotą jest interakcja typu człowiek-system-człowiek (zob. tabela 3.1, str. 42). Tym bardziej może być wykorzystywana w mniej rozbudowanych procesach, w których liczba dokumentów i pracowników jest stosunkowo niewielka.

Stąd też jako środowisko biznesowe do walidacji omawianej architektury wybrałam proces zarządzania przedmiotem w szkole wyższej. Jest to problem dobrze mi znany, a ponadto występują w nim wszystkie elementy pozwalające na walidację architektury MIND. Decyzje dotyczące ocen są podejmowane przez pracowników, często w oparciu o ich własną wiedzę, zatem proces oceniania jest w ogólności procesem niealgorytmicznym. Dotyczy to szczególnie oceny odpowiedzi na pytania otwarte czy weryfikację samodzielności wykonania zadań przez ocenianego studenta. W procesie zarządzania przedmiotem występują też wszystkie sytuacje opisane wzorcami koordynacji dokumentu (podrozdział 5.3), przez co wybrane studium przypadku jest wystarczające do weryfikacji implementowalności tych wzorców.

Opis i diagramy przepływu pracy procesu zarządzania przedmiotem, zawierającego wszystkie wzorce koordynacji dokumentu, znajdują się w załączniku C.

Przykładowy przepływ dokumentów w systemie zarządzania przedmiotem został przedstawiony na rysunku 6.8 – celem jego działania jest zebranie i archiwizacja ocen studentów z danego przedmiotu. Proces rozpoczyna się w dziekanacie, gdzie tworzony jest dokument zawierający listę studentów przypisanych do danego przedmiotu wraz z przydziałem do grup i wyszczególnieniem rodzaju zajęć. Grupy studentów mogą się różnić w zależności od rodzaju zajęć, np. w grupie wykładowej są jednocześnie wszyscy studenci, w grupach ćwiczeniowych jest po 30 studentów, zaś w grupach laboratoryjnych po 20. Zatem każdy student ma przypisaną odpowiednią grupę. Dołączany jest również rozkład zajęć, określający np. liczbę godzin poszczególnych zajęć w tygodniu.



Rys. 6.8. Przykład procesu zarządzania przedmiotem na wyższej uczelni

Pracowników wiedzy, biorących udział w procesie zarządzania przedmiotem, można podzielić na czynnych i biernych. Biernymi uczestnikami są pracownicy dziekanatu, którzy tworzą rdzeń dokumentu, ale nie mają uprawnień do edycji dokumentów składowych. Biernymi uczestnikami mogą być dodatkowo studenci – nie ujęci na rysunku 6.8, ale mogą otrzymać prawo do wglądu we własne oceny na dowolnym etapie ich wystawiania. Pracownikami czynnymi, czyli decydentami wystawiającymi oceny, są nauczyciele prowadzący poszczególne części składowe przedmiotu, a więc wykład, ćwiczenia, laboratoria itp. oraz kierownik przedmiotu.

Dokument MIND dla procesu zarządzania przedmiotem zawiera następujące

dane:

- komponent główny zawiera unikatowy identyfikator dokumentu, a także nazwę przedmiotu, dane pracownika inicjującego dokument oraz kierownika przedmiotu oraz datę utworzenia. Dodatkowo, może zawierać informacje o przedmiocie: dla jakiego kierunku czy specjalności jest przeznaczony, na którym semestrze się odbywa itp.;

- komponent pracowników wiedzy zawiera dane personalne i kontaktowe osób biorących udział w obiegu dokumentu: pracownika dziekanatu, kierownika przedmiotu, prowadzących poszczególne zajęcia;

- komponent dokumentów składowych przechowuje informacje na temat list poszczególnych grup. Początkowo, opisuje jeden dokument, czyli ogólną listę wszystkich studentów przydzielonych do przedmiotu. Listy poszczególnych grup będą tworzone automatycznie na podstawie danych zawartych w dokumencie, w którym studenci są przypisani do konkretnej grupy;

- komponent usług opisuje usługi potrzebne do transformacji treści dokumentów składowych, a zatem aplikacje pozwalające na ich prawidłowe odtworzenie oraz transformacje pozwalające na rozdzielenie dokumentu na grupy i łączenie rozdzielonych dokumentów w jedną listę.

- komponent przepływu pracy zawiera definicję struktury procesu, w której mogą się znaleźć wszystkie wzorce koordynacji dokumentu.

Wzorzec sekwencyjny

Wzorzec sekwencyjnego przepływu dokumentów dotyczy sytuacji, gdy dokument z dziekanatu trafia do kierownika przedmiotu, a po zebraniu wszystkich ocen cząstkowych i wystawieniu ocen końcowych, lista z powrotem jest przekazywana do dziekanatu. Przepływ sekwencyjny może występować też w innych sytuacjach w procesie, np. gdy zajęcia w danej grupie prowadzą kolejno różni nauczyciele. Dokument może być przesyłany w kliku mniejszych częściach, gdy jego rozmiar był zbyt duży, by skutecznie przesłać go poprzez sieć. Może też być przesłany w kilku wersjach, np. gdy nauczyciel prowadzący pewien rodzaj zajęć przesyła listę ocen studentów z zaznaczeniem, że nie jest to jeszcze wersja kompletna, zaś wersja ostateczna jest oznaczana przez znacznik końca. W pierwszym przypadku, dokument po odebraniu wszystkich części jest scalany, zaś w drugim przypadku tylko ostateczna wersja zostaje zaakceptowana.

Wzorzec rozdzielający dokument

Różne rodzaje zajęć w poszczególnych grupach są często prowadzone przez różnych nauczycieli. Dokument z listą studentów przedmiotu jest wówczas rozdzielany na odpowiednie części składowe i każdy z pracowników otrzymuje listy studentów tylko tych grup, które prowadzi. Przepływ dokumentów składowych z listami grup odbywa się wówczas równolegle. Wersja kopiująca wzorca odpowiada natomiast sytuacji, w której praca studentów (np. dokumentacja projektu) jest przekazywana równolegle do kilku recenzentów.

Wzorzec łączący dokument

Po wystawieniu ocen z danego rodzaju zajęć, części składowe dokumentu wracają do kierownika przedmiotu wraz z załącznikami, jeśli były wymagane. W zależności od wersji wzorca, kierownik przedmiotu może wystawić oceny końcowe zaraz po otrzymaniu listy danej grupy lub po wpłynięciu i scaleniu wszystkich oczekiwanych części dokumentu

Wzorzec dyskryminującego łączenia w procesie oceniania wystąpi, gdy przy ocenie istotna będzie kolejność zgłoszeń. Przykładowo, dodatkowe punkty otrzyma student, który rozwiąże zadany problem jako pierwszy, mimo tego proces nadal czeka na rozwiązania innych studentów lub na dany termin kolokwium może zapisać się tylko pewna określona liczba studentów, ale pozostali też zostają zapisani na inny termin.

Wzorzec iteratora

Wzorzec iteratora może wystąpić w sytuacji, w której kierownik przedmiotu otrzymuje listę ocen po wystawieniu każdej oceny cząstkowej przez prowadzącego zajęcia i odsyła tę listę z powrotem przed kolejnymi zajęciami lub w przypadku konieczności naniesienia poprawek. Inną sytuacją występowania tego wzorca jest, gdy kilku nauczycieli prowadzi zamiennie zajęcia w danej grupie i po każdych zajęciach przekazują sobie listę studentów tej grupy. Powstają w ten sposób dowolne niestrukturalne cykle obiegu dokumentów.

Wzorzec odroczonego wyboru

W procesie zarządzania przedmiotem może zdarzyć się sytuacja, w której pracownik w danej chwili nie wie, komu przekazać dokument. Taka sytuacja może wystąpić, gdy prowadzący zajęcia z projektu grupowego ma przesłać dokumentację projektu do recenzentów. Recenzentów zaś wyznacza kierownik przedmiotu i to od jego decyzji zależy, komu prowadzący prześle dokumentację.

Wzorzec kamienia milowego

Wzorzec ten możliwości wykonania pewnych czynności w zależności od stanu realizacji innych czynności. Przykładem takiej sytuacji jest możliwość poprawy ocen przez studentów, dopóki lista ocen nie zostanie zatwierdzona w dziekanacie.

Wzorzec anulujący

Anulowanie czynności związane jest zazwyczaj z jakimiś sytuacjami wyjątkowymi. Przykładowo, czynność wystawiania ocen na danych zajęciach jest anulowana z powodu ogłoszenia godzin dziekańskich albo kolokwium zostaje anulowane, z powodu wykrycia nieprawidłowości w jego przebiegu.

Wewnętrzny podproces

Często, kierownik przedmiotu posiada informacje, które nie są znane w dziekanacie, np. o pracownikach prowadzących poszczególne części czy grupy przedmiotu. Może więc zmodyfikować ścieżkę migracji, chcąc przekazać poszczególne części dokumentu do odpowiednich pracowników. Wówczas proces nadrzędny ma postać prostego przepływu od pracownika dziekanatu do kierownika przedmiotu i z powrotem, natomiast kierownik przedmiotu, w ramach czynności zbierania ocen, definiuje podproces obiegu dokumentów między poszczególnymi nauczycielami.

Zewnętrzny podproces

Inną sytuacją jest, gdy część zajęć z przedmiotu lub cały przedmiot jest prowadzony przez inną katedrę lub wydział, a w przypadku niektórych zajęć projektowych lub laboratoryjnych, także przez osoby z firmy współpracującej z uczelnią. Gdy zajęcia są zlecane zewnętrznemu wykonawcy, ich proces realizacji może być nieznany zleceniodawcy. Efektem realizacji procesu zewnętrznego jest dostarczenie gotowej listy ocen studentów lub recenzji.

Rozdział 7

Podsumowanie

7.1. Realizacja i weryfikacja tezy

Wynikiem rozprawy jest model architektury mobilnych dokumentów interaktywnych MIND, który został opracowany w celu efektywnego zarządzania informacją w procesach podejmowanych przez organizacje oparte na wiedzy. Architektura ta stanowi całościowy opis struktury i funkcjonalności dokumentu, zaprojektowanego w celu realizacji rozproszonego przetwarzania zespołowego i pracy grupowej. Umożliwia automatyczne przejście od statycznej postaci dokumentów XML do dynamicznej postaci obiektowej. Obiekty MIND mają cechy mobilnych agentów, gdyż potrafią samodzielnie przemieszczać się między urządzeniami uczestników procesu oraz autonomicznie wykonywać swoją wbudowaną funkcjonalność.

Celem architektury jest, zgodnie z tezą rozprawy, **umożliwienie efektywnego zarządzania informacją w dynamicznych wirtualnych organizacjach przy rozwiązywaniu złożonych, niealgorytmicznych problemów decyzyjnych.** Charakterystyka wirtualnych organizacji ludzkich oraz złożoności podejmowanych przez nie niealgorytmicznych procesów decyzyjnych została dokładnie omówiona w rozdziałach 2 i 3 rozprawy. Zgodnie z charakterystyką problemów rozwiązywanych przez te organizacje, celem architektury MIND nie jest automatyzacja procesów decyzyjnych, a zarządzanie ich realizacją w interakcji z człowiekiem.

Cel ten został zrealizowany poprzez ułatwienie *komunikacji treści* zawartych w przesyłanych dokumentach oraz umożliwienie *koordynacji czynności* procesów podejmowanych przez ludzi. Komunikacja treści, realizowana przez agenty MIND, dotyczy możliwości przekazywania dokumentów między rozproszonymi uczestnikami procesu (pracownikami wiedzy) oraz wsparcia prawidłowego przetwarzania zawartości różnego typu dokumentów na urządzeniach pracowników. Agenty MIND koordynują wykonywanie czynności procesu poprzez wbudowaną politykę migracji oraz

możliwość realizacji kanonicznego zbioru wzorców koordynacji dokumentu (przedstawionego w podrozdziale 5.3).

Teza rozprawy została zweryfikowana poprzez wykazanie realizowalności architektury MIND jako otwartego systemu agentowego zgodnego z modelem architektury 3-warstwowej (rys. 6.1), w którym dokumenty-agenty MIND stanowią warstwę pośredniczącą. W szczególności, mobilność i wykonywalność agentów-dokumentów MIND została bezpośrednio wykazana za pomocą dwóch prototypowych systemów agentowych: systemu zaimplementowanego w oparciu o dojrzałą platformę agentową JADE oraz luźno powiązanego systemu (ang. *ang. loosely coupled system*), wykorzystującego pocztę elektroniczną jako warstwę transportową. Implementacja kanonicznych wzorców koordynacji dokumentu w środowisku wytwarzania przepływów pracy Enhydra Shark i bezpośrednio w architekturze MIND za pomocą dwóch prototypów systemu obiegu determinuje z kolei realizowalność procesów wiedzy w organizacjach ludzkich.

7.2. Nowatorstwo rozwiązań

Oryginalnym rozwiązaniem, przedstawionym w rozprawie, jest przede wszystkim architektura MIND (rozdział 4). Stanowi ona model przetwarzania obiektowego skoncentrowanego na dokumencie (ang. *document-centric*), w którym komponenty dokumentu stanowią jednocześnie jednostkę informacji i interfejsu. Główne cechy komponentów MIND to:

- *mobilność*, czyli możliwość samodzielnego przemieszczania się między rozproszonymi jednostkami obliczeniowymi zgodnie z wbudowaną polityką migracji,
- *dynamizm*, czyli zdolność do autonomicznej realizacji swojej funkcjonalności,
- *interaktywność*, czyli zdolność uruchamiania odpowiednich usług pośredniczących w przetwarzaniu treści dokumentów (automatycznie lub przez człowieka).

Kolejnym rozwiązaniem jest propozycja kanonicznego zbioru wzorców koordynacji dokumentu (rozdział 5), opracowanego specjalnie dla dokumentów MIND na podstawie wzorców przepływu sterowania ogólnego podejścia procesowego opisanego w literaturze [ARH11].

W rozprawie został też zaproponowany model organizacji opartej na wiedzy (rozdział 2), który pozwala na skonstruowanie procesu wiedzy realizowanego przez dokument MIND.

W szczególności, koncepcja usługi rozproszonego zarządzania przepływem pracy (rozdział 5) w oparciu o dane dotyczące procesu przynoszone przez dokument MIND

jest nowa, wcześniej nie rozważana w literaturze jako immanentna cecha dokumentów proaktywnych. Koncepcję tę realizuje aplikacja LWE, która pełni rolę „lekkiego" klienta pocztowego i umożliwia realizację wykonywalności i mobilności komponentów MIND. Idea takiego klienta też jest nowa i stanowi interesującą alternatywę dla rozwiązania problemów warstwy transportowej w systemach agentowych

Dla potrzeb walidacji architektury, powstały dwa środowiska agentowe realizujące mobilność i funkcjonalność dokumentów MIND: w oparciu o platformę JADE i system klientów LWE, został także zrealizowany zbiór wzorców koordynacji dokumentu w oparciu o system Enhydra Shark (rozdział 6).

7.3. Perspektywy rozwoju

Rozprawa nie podejmuje dwóch kwestii, jakie zwykle wiążą się z aplikacjami przetwarzania rozproszonego, tj. bezpieczeństwa i niezawodności. Celem rozprawy było przede wszystkim wykazanie możliwości realizacji procesów w środowisku rozproszonym w interakcji z ludźmi, którą zapewnia mobilność i wykonywalność komponentów dokumentu. Bezpieczeństwo i niezawodność wiążą się raczej z jakością realizacji procesów, a nie z ich realizowalnością. Niemniej jednak, kwestie te stanowią temat moich dalszych badań nad architekturą MIND, które niżej tylko zasygnalizuję.

W procesach realizowanych w oparciu o rozproszony dokument mogą zdarzyć się dwie niepożądane sytuacje: niektóre komponenty dokumentu mogą zaginąć lub komponent może trafić do nieodpowiedniego użytkownika (przypadkowo lub na skutek zaplanowanego przechwycenia). Aby przeciwdziałać tym sytuacjom, zostały zaproponowane dwie nowe polityki: *kompletności* i *bezpieczeństwa*. Możliwość rozbudowy architektury MIND o nowe polityki została przedstawiona na rysunku 1.2. Lista polityk nie jest zamknięta, kolejnym elementem jest polityka negocjacji, rozszerzająca funkcjonalność dokumentu o zdolność do uczenia się w celu rozpoznawania dynamicznego kontekstu wykonywania czynności [Kac].

Polityka kompletności wiąże się ze śledzeniem lokalizacji komponentów dokumentu w trakcie realizacji procesu. W tym celu proponowane jest wykorzystanie *Handle System* [SLB03], będącego systemem odpowiedzialnym za zapewnienie trwałości identyfikatorów obiektów cyfrowych, w tym dokumentów elektronicznych. Jednym z przykładów jego zastosowania jest identyfikator DOI [Pas12], stosowany powszechnie w bibliotekach cyfrowych. Trwałe identyfikatory jednoznacznie identyfikują dokumenty, niezależnie od ich fizycznego umiejscowienia i są powiązane z bazą danych rejestrującą zmiany lokalizacji dokumentów. Dzięki temu rozwiązaniu, można odnaleźć dokument, niezależnie od jego aktualnej lokalizacji w przestrzeni Internetu.

Polityka zabezpieczenia treści dokumentu wiąże się, natomiast, z uniemożliwie-

niem dostępu do zawartości dokumentu użytkownikom, którzy nie są przypisani do danej czynności. W tym celu stosuje się opracowane w Katedrze Inteligentnych Systemów Interaktywnych (WETI) algorytmy biometrycznego zabezpieczenia treści [SSW13]. Pozwalają one na autoryzację użytkownika na podstawie jego cech biometrycznych (twarzy, sposobu pisania na klawiaturze czy posługiwania się myszką). Dokument zabezpieczony biometrycznie, dzięki swojej funkcjonalności, może samodzielnie dokonywać uwierzytelniania użytkownika. Takie rozwiązanie eliminuje konieczność ustalania silnych haseł, co jest często trudne i kłopotliwe dla pracowników organizacji.

Wspomniane wyżej polityki są aktualnie opracowywane w ramach projektu badawczego MENAID [MeN13], w którym uczestniczę jako główny wykonawca.

Podziękowania

Chciałabym w szczególności podziękować mojemu promotorowi, prof. dr hab. inż. Bogdanowi Wiszniewskiemu za czas spędzony podczas konsultacji, za pomysłowość popartą wiedzą i za motywowanie mnie do twórczej pracy.

Pragnę podziękować także mojemu mężowi Wojtkowi, rodzicom, teściom, rodzinie i znajomym za ogromne wsparcie, jakie mi okazali w trakcie mojej pracy nad rozprawą.

Składam podziękowania Narodowemu Centrum Nauki za uznanie inżynierii dokumentu jako dyscypliny leżącej w zakresie badań podstawowych i za przyznanie finansowania dla projektu badawczego nr 2011/01/B/ST6/06500, w ramach którego mogłam kontynuować pracę nad architekturą dokumentu MIND

Moje badania wspierane były także grantem „InnoDoktorant - stypendia dla doktorantów III ed.". w ramach projektu współfinansowanego przez Unię Europejską z Europejskiego Funduszu Społecznego (Program Operacyjny Kapitał Ludzki, Priorytet VIII, Działanie 8.2, Poddziałanie 8.2.2: „Regionalne Strategie Innowacji"). Dodatkowe wsparcie otrzymałam również od Pomorskiej Specjalnej Strefy Ekonomicznej, która przyznała mi stypendium w ramach realizacji projektu „InnoDoktorant - stypendia dla doktorantów II ed."

Bibliografia

[Aal03] Wil M.P. van der Aalst. Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language. Raport techniczny FIT-TR-2003-06, Queensland University of Technology, Brisbane, 2003.

[Ado04] Adobe Systems Incorporated. *PDF Reference Version 1.6 (5th Edition).* Adobe Press, 2004.

[Ado13] Adobe Acrobat Family. Adobe Reader XI. http://www.adobe.com/pl/products/reader.html, 2013.

[Adv09] Advanced Distributed Learning. SCORM 2004 Content Aggregation Model. Standard, 2009.

[AHKB03] Wil M.P. van der Aalst, Arthur H.M. ter Hofstede, Bartek Kiepuszewski, Alistair P. Barros. Workflow Patterns. *Distrib. Parallel Databases*, 14(1):5–51, 2003.

[Alt13] Altova. XMLSpy. http://www.altova.com/xmlspy.html, 2013.

[AMRZ01] Sharon Adler, Alex Milowski, Jeremy Richman, Steve Zilles. Extensible Stylesheet Language (XSL) - Version 1.1, 2001.

[Apa04] Apache Software Foundation. The Apache XML Project. http://xmlbeans.apache.org/sourceAndBinaries/, 2004.

[Apa12] Apache Software Foundation. Logging Services. http://logging.apache.org/log4j/1.2/, 2012.

[ARH11] Wil M.P. van der Aalst, Nick Russell, Arthur H.M. ter Hofstede. Workflow Patterns Home Page. http://www.workflowpatterns.com/patterns/, 2011.

[BCG07] Fabio Luigi Bellifemine, Giovanni Caire, Dominic Greenwood. *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology).* John Wiley & Sons, 2007.

[BDH+05] Victoria Bellotti, Nicolas Ducheneaut, Mark Howard, Ian Smith, Rebecca E. Grinter. Quality versus quantity: e-mail-centric task management and its relation with overload. *Hum.-Comput. Interact.*, 20(1):89–138, 2005.

[Bel73] Daniel Bell. *The Coming of Post-Industral Society.* Basic Books, 1973.

[BL08] Michel Beaudouin-Lafon. Interaction is the Future of Computing. Thomas Erickson, David McDonald (red.), *HCI Remixed, Reflections on Works That Have Influenced the HCI Community*, strony 263–266. MIT Press, 2008.

[Bou11] Ronald Bourret. XML Data Binding Resources. http://www.rpbourret.com/xml/XMLDataBinding.htm, 2011.

[BPSM+08] Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, François Yergeau. Extensible Markup Language (XML) 1.0 (Fifth Edition). World Wide Web Consortium, Recommendation REC-xml-20081126, 2008.

[BS11] Gregory R. Ball, Sargur N. Srihari. Statistical characterization of handwriting characteristics using automated tools. *DRR*, strony 1–10, 2011.

[Bug13] Ryszard Bugajski. Układ zamknięty. Film, Filmicon Dom Filmowy s.c., http://ukladzamkniety.com/, 2013.

[Cas10] Liane Cassavoy. Take Control of Outlook E-Mail With NEO Pro. http://www.pcworld.com/article/212524/neo_pro.html, 2010.

[CGMS94] Nicholas J. Carriero, David Gelernter, Timothy G. Mattson, Andrew H. Sherman. The Linda alternative to message-passing systems. *Parallel Comput.*, 20(4):633–655, 1994.

[Chr06] Per Christensson. The Tech Terms Computer Dictionary. http://www.techterms.com/, 2006.

[CHSD12] Pranob K. Charles, V. Harish, M. Swathi, CH. Deepthi. A Review on the Various Techniques used for Optical Character Recognition. *International Journal of Engineering Research and Applications (IJERA)*, strony 659–662, 2012.

[Cri03] M. Crispin. Internet Message Access Protocol - Version 4rev1. Internet RFC 3501, http://tools.ietf.org/html/rfc3501, 2003.

[Cro06] Douglas Crockford. The application/json Media Type for JavaScript Object Notation (JSON). Internet RFC 4627, http://tools.ietf.org/html/rfc4627, 2006.

[CTZ02] Paulo Ciancarini, Robert Tolksdorf, Franco Zambonelli. A survey of coordination middleware for XML-centric applications. *Knowl. Eng. Rev.*, 17:389–405, 2002.

[Czy13] Mariusz Czyż. Klient inteligentnej poczty elektronicznej. Praca inżynierska, Wydział Elektroniki, Telekomunikacji i Informatyki, Politechnika Gdańska, 2013.

[Deb11] Nicolas Debeissat. Converting XSD schema to RelaxNG. http://debeissat.nicolas.free.fr/XSDtoRNG.php, 2011.

[DEL+00] Paul Dourish, W. Keith Edwards, Anthony LaMarca, John Lamping, Karin Petersen, Michael Salisbury, Douglas B. Terry, James Thornton. Extending document management systems with user-specific active properties. *ACM Trans. Inf. Syst.*, 18(2):140–170, 2000.

[DK06] Laura A. Dabbish, Robert E. Kraut. Email overload at work: an analysis of factors associated with email strain. *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, CSCW'06, strony 431–440, New York, USA, 2006. ACM.

[Dou03] Paul Dourish. The Appropriation of Interactive Technologies: Some Lessons from Placeless Documents. *Comput. Supported Coop. Work*, 12(4):465–490, 2003.

[FB96] Ned Freed, Nathaniel Borenstein. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. Internet RFC 2045, http://www.ietf.org/rfc/rfc2045.txt, 1996.

[FC04] Alain Frisch, Luca Cardelli. Greedy Regular Expression Matching. Josep Díaz, Juhani Karhumäki, Arto Lepistö, Donald Sannella (red.), *ICALP*, Lecture Notes in Computer Science 3142, strony 618–629. Springer, 2004.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Internet RFC 2616, http://www.w3.org/Protocols/rfc2616/rfc2616.html, 1999.

[FK05] Ned Freed, John C. Klensin. Media Type Specifications and Registration Procedures. Internet RFC 4288, http://tools.ietf.org/html/rfc4288, 2005.

[Fou02] Foundation for Intelligent Physical Agents. FIPA Abstract Architecture Specification. http://www.fipa.org, 2002.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Design patterns: elements of reusable object-oriented software.* Addison-Wesley Longman Publishing Co., Inc., Boston, USA, 1995.

[GM05] Robert J. Glushko, Tim McGrath. Document Engineering: analyzing and designing the semantics of Business Service Networks. *Proceedings of the IEEE EEE05 international workshop on Business services networks*, BSN '05, Piscataway, NJ, USA, 2005. IEEE Press.

[GM08] Robert J. Glushko, Tim McGrath. *Document Engineering - Analyzing and Designing Documents for Business Informatics and Web Services.* MIT Press, 2008.

[God06a] Magdalena Godlewska (Łabiak). Model dokumentu wielowarstwowego PDF. *Technologie informacyjne, IV Krajowa Konferencja Technologie Informacyjne*, Gdańsk, 2006. Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki.

[God06b] Magdalena Godlewska (Łabiak). Rozproszone dokumenty elektroniczne. *Seminarium Naukowo - Techniczne TECHNICON'06*, strony 1–11, Gdańsk, 2006. Politechnika Gdańska, Wydział Mechaniczny.

[God08] Magdalena Godlewska. Completeness and Consistency of Dostributed Electronic Documents. *Conference Human System Interaction HSI'08*, strony 1–4, Kraków, 2008. Wyższa Szkoła Informatyki i Zarządzania w Rzeszowie.

[God09] Magdalena Godlewska. Otwarta architektura dokumentu wspierającego podejmowanie decyzji w organizacjach opartych na wiedzy. Zdzisław Kowalczuk (red.), *Systemy wykrywające, analizujące i tolerujące usterki*, strony 235–242, Gdańsk, 2009. Pomorskie Wydawnictwo Naukowo-Techniczne PWNT (Automatyka i Informatyka: Technologie Informacyjne Automatyka i Diagnostyka; Nr 6).

[God10a] Magdalena Godlewska. Agent System for Managing Distributed Mobile Interactive Documents. Piotr Jędrzejowicz, i in. (red.), *Agent and Multi-Agent Systems: Technologies and Applications, 4th KES International Symposium, KES-AMSTA Gdynia 2010*, strony 390–399, Berlin, 2010. Springer-Verlag, LNAI 6071.

[God10b] Magdalena Godlewska. Workflow Patterns Applicable to Virtual Knowledge-Based Organizations. Alicja Konczanowska, Lech Hasse (red.), *Information Technologies, 2nd International Conference on Information Technology*, wol. 18, strony 33–38, Gdańsk, 2010.

[God12] Magdalena Godlewska. Agent System for Managing Distributed Mobile Interactive Documents in Knowledge-Based Organizations. Ngoc Thanh Nguyen (red.), *Transactions on Computational Collective Intelligence VI*, LNCS 7190, strony 121–145, Berlin, 2012. Springer-Verlag.

[Goo12] Google. Google Drive. https://drive.google.com, 2012.

[Gre07] Kate Greene. A New Look for Outlook. http://www.technologyreview.com/news/408772/a-new-look-for-outlook/, 2007.

[Gre11] Samuel Greengard. Living in a digital world. *Commun. ACM*, 54(10):17–19, 2011.

[GSW04] Dina Q. Goldin, Scott A. Smolka, Peter Wegner. Turing Machines, Transition Systems, and Interaction. *Information and Computation*, 194:101–128, 2004.

[GW05] Dina Goldin, Peter Wegner. The Church-Turing Thesis: breaking the myth. *Proceedings of the First international conference on Computability in Europe: new Computational Paradigms*, CiE'05, strony 152–168, Berlin, Heidelberg, 2005. Springer-Verlag.

[GW08] Dina Goldin, Peter Wegner. The Interactive Nature of Computing: Refuting the Strong Church-Turing Thesis. *Minds Mach.*, 18(1):17–38, 2008.

[GW09] Magdalena Godlewska, Bogdan Wiszniewski. Architektura MIND Mobilnych Interaktywnych Dokumentów rozproszonych. Raport techniczny 18/2009 WETI - 2009, Wydział Elektroniki, Telekomunikacji i Informatyki, Politechnika Gdańska, 2009.

[GW10] Magdalena Godlewska, Bogdan Wiszniewski. Distributed MIND - a new processing model based on mobile interactive documents. Roman Wyrzykowski, i in. (red.), *Parallel Processing and Applied Mathematics, 8th International Conference, PPAM 2009*, strony 244–249, Berlin, 2010. Springer-Verlag, LNCS 6068.

[HA05] Arthur H.M. ter Hofstede, Wil M.P. van der Aalst. YAWL: yet another workflow language. *Information Systems 30(4)*, strony 245–275, 2005.

[JAD13] JADE. Forum internetowe, http://jade.17737.x6.nabble.com/, 2013.

[Jen87] Kurt Jensen. Coloured Petri Nets. *Petri Nets: Central Models and Their Properties*, strony 248–299, 1987.

[JK09] Kurt Jensen, Lars M. Kristensen. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. Springer-Verlag, Berlin, 2009.

[Kac] Jerzy Kaczorek. *Metody automatycznego negocjowania uzgodnień w realizacji usług internetowych (w przygotowaniu)*. Praca doktorska, Wydział Elektroniki, Telekomunikacji i Informatyki, Politechnika Gdańska.

[Kan97] Kancelaria Sejmu RP. Kodeks postępowania karnego. Ustawa, http://isap.sejm.gov.pl/DetailsServlet?id=WDU19970890555, 1997.

[Kle08] John C. Klensin. Simple Mail Transfer Protocol. Internet RFC 5321, https://tools.ietf.org/html/rfc5321, 2008.

[KW98] Henryk Krawczyk, B. Wiszniewski. *Analysis and Testing of Distributed Software Applications*. Taylor & Francis, Inc., Bristol, PA, USA, 1998.

[LED+99] Anthony Lamarca, W. Keith Edwards, Paul Dourish, John Lamping, Ian Smith, Jim Thornton. Taking the Work out of Workflow: Mechanisms for Document-Centered Collaboration. *Proceedings of the European Conf. Computer-Supported Cooperative Work ECSCW'99*, strony 1–20. Kluwer, 1999.

[LK01] Agnieszka Lekka-Kowalik. Ukryte założenia idei społeczeństwa informacyjnego. Tadeusz Zasępa (red.), *Internet - fenomen społeczeństwa informacyjnego*. Edycja Świętego Pawła, 2001.

[LR05] Tancred Lindholm, Torsten Rüger. A Fault-Tolerant Three-Way Merge for XML and HTML. M. H. Hamza (red.), *Internet and Multimedia Systems and Applications, EuroIMSA 2005, Grindelwald, Switzerland, February 21-23, 2005*, strony 71–76. IASTED/ACTA Press, 2005.

[Mac10] Simon Mackie. ClearContext Personal Makes Outlook More Productive. http://gigaom.com/2010/11/16/clearcontext-personal-makes-outlook-more-like-gmail/, 2010.

[MCZE02] Cecilia Mascolo, Licia Capra, Stefanos Zachariadis, Wolfgang Emmerich. XMIDDLE: A Data-Sharing Middleware for Mobile Computing. *Int. Journal on Personal and Wireless Communications*, 2002.

[MeN13] MeNaID. MEtody i NArzędzia Inżynierii Dokumentu przyszłości. http://menaid.org.pl/, 2013.

[Mes12] MesQuilla. TaQuilla. http://mesquilla.com/extensions/taquilla/, 2012.

[Mic07] Microsoft. SkyDrive - Window Live. https://skydrive.live.com, 2007.

[MiG11] MiG InfoCom. MigLayout - Java Layout Manager for Swing, SWT and JavaFX 2. http://www.miglayout.com/, 2011.

[Mik06] Bogusz Mikuła. *Organizacje oparte na wiedzy.* WAEK, Kraków, 2006.

[MLMK05] Makoto Murata, Dongwon Lee, Murali Mani, Kohsuke Kawaguchi. Taxonomy of XML schema languages using formal language theory. *ACM Trans. Internet Technol.*, 5(4):660–704, Listopad 2005.

[MPR06] Amy L. Murphy, Gian Pietro Picco, Gruia-Catalin Roman. LIME: A coordination model and middleware supporting mobility of hosts and agents. *ACM Transactions on Software Engineering and Methodology*, 15:279–328, 2006.

[MR96] J. Myers, M. Rose. Post Office Protocol - Version 3. Internet RFC 1939, http://www.ietf.org/rfc/rfc1939.txt, 1996.

[MWL08] Daniel Martin, Daniel Wutke, Frank Leymann. Synchronizing control flow in a tuplespace-based, distributed workflow management system. *Proceedings of the 10th international conference on Electronic commerce*, ICEC '08, strony 11:1–11:9, New York, NY, USA, 2008. ACM.

[Obj11] Object Management Group. Business Process Model and Notation (BPMN). Raport techniczny formal/2011-01-03, Object Management Group, January 2011.

[Obr12] Karol Obrębski. Absurdy prawa: co z tym e-sądem? http://www.forbes.pl/e-sad-wzywa-do-zaplaty-nieistniejacej-naleznosci,artykuly,135507,1,1.html, 2012.

[Ora10] Oracle. Javadoc - The Java API Documentation Generator. http://docs.oracle.com/javase/1.5.0/docs/tooldocs/windows/javadoc.html, 2010.

[Ora13a] Oracle. Balloon tip. https://balloontip.java.net/, 2013.

[Ora13b] Oracle. Java Mail. http://www.oracle.com/technetwork/java/javamail/index.html, 2013.

[Ora13c] Oracle Corporation. JAXB Release Documentation. https://jaxb.java.net/2.2.6/docs, 2013.

[Org01] Organization for the Advancement of Structured Information Standards. RELAX NG Specification. https://www.oasis-open.org/committees/relax-ng/spec-20011203.html, 2001.

[Org02] Organization for the Advancement of Structured Information Standards. RELAX NG Compact Syntax. https://www.oasis-open.org/committees/relax-ng/compact-20021121.html, 2002.

[OUM+05] Gérald Oster, Pascal Urso, Pascal Molli, Hala Skaf-Molli, Abdessamad Imine. Optimistic Replication for Massive Collaborative Editing. Raport techniczny RR-5719, INRIA, 2005.

[OUMI05] Gérald Oster, Pascal Urso, Pascal Molli, Abdessamad Imine. Proving correctness of transformation functions in collaborative editing systems. Raport techniczny RR-5795, INRIA, 2005.

[Pas12] Norman Paskin. *DOI Handbook*. International DOI Foundation, 2012.

[Phe05] Thomas A. Phelps. Multivalent Home Page. http://multivalent.sourceforge.net/, 2005.

[PW96] Thomas A. Phelps, Robert Wilensky. Toward active, extensible, networked documents: multivalent architecture and applications. *In Proceedings of the 1st ACM International Conference on Digital Libraries*, strony 100–108, 1996.

[PW98] Thomas A. Phelps, Robert Wilensky. Multivalent Documents: A New Model for Digital Documents. Raport techniczny UCB/CSD-98-999, EECS Department, University of California, Berkeley, 1998.

[Res00] E. Rescorla. HTTP Over TLS. Internet RFC 2818, http://www.rfc-base.org/txt/rfc-2818.txt, 2000.

[RHAM06] Nick Russell, Arthur H.M. ter Hofstede, Wil M.P. van der Aalst, Nataliya Mulyar. Workflow Control-Flow Patterns: A Revised View, 2006.

[Rob07] Wojciech Robaczyński. Błąd w sztuce medycznej jako podstawa odpowiedzialności odszkodowawczej. http://www.abc.com.pl/problem/143/7, 2007.

[Sat01] Ichiro Satoh. Mobile agent-based compound documents. *Proceedings of the 2001 ACM Symposium on Document engineering*, DocEng '01, strony 76–84, New York, USA, 2001. ACM.

[Sch12] Christian Schenk. MikTex 2.9 manual. http://docs.miktex.org/manual/, 2012.

[Sha13] ShareLaTeX. Real Time LaTeX Collaboration. https://www.sharelatex.com/, 2013.

[Sic08] Jacek Siciarek. Środowisko narzędziowe do wytwarzania inteligentnych dokumentów elektronicznych. Praca magisterska, Wydział Elektroniki, Telekomunikacji i Informatyki, Politechnika Gdańska, 2008.

[SLB03] S. Sun, M Lannom, B Boesch. Handle System Overview. Internet RFC 3650, http://www.handle.net/rfc/rfc3650.html, 2003.

[Sos11] Dennis Sosnoski. JiBX: Binding XML to Java Code. http://jibx.sourceforge.net/index.html, 2011.

[Spi11] J.B. Spira. *Overload! How Too Much Information is Hazardous to your Organization.* Wiley, 2011.

[SSW13] Jacek Siciarek, Maciej Smiatacz, Bogdan Wiszniewski. For Your Eyes Only – Biometric Protection of PDF Documents. *EEE'13 - The 2013 International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government*, (przyjęty do druku <22-25.07.2013>), Las Vegas, USA, 2013.

[Sta87] Peter H. Starke. *Sieci Petri - podstawy, zastosowania, teoria.* PWN, Warszawa, 1987.

[Szc08] Jarosław Szczepański. Serwer usług bazowych do zarządzania konfiguracją inteligentnego dokumentu elektronicznego. Praca magisterska, Wydział Elektroniki, Telekomunikacji i Informatyki, Politechnika Gdańska, 2008.

[Tel13] Telecom Italia. Workflows and Agents Development Environment, an Open Source Platform for Workflows and Agents based applications. http://jade.tilab.com/wade/, 2013.

[Tha08] Thai Open Source Software Center Ltd. Trang - Multi-format schema converter based on Relax NG. http://www.thaiopensource.com/relaxng/trang.html, 2008.

[TIM07] Raquel Trillo, Sergio Ilarri, Eduardo Mena. Comparison and Performance Evaluation of Mobile Agent Platforms. *Proceedings of the Third International Conference on Autonomic and Autonomous Systems*, ICAS '07, Washington, USA, 2007. IEEE Computer Society.

[TNSV11] Tõnu Tamme, Ulrich Norbisrath, Georg Singer, Eero Vainikko. Improving Email Management. *IMMM 2011, The First International Conference on Advances in Information Mining and Management*, strony 67 – 72, 2011.

[Tog11a] Together Teamsolutions Co. Together Workflow Server: User Manual, 2011.

[Tog11b] Together Teamsolutions Co. Together XPDL and BPMN Workflow Editor. http://www.together.at/prod/workflow/twe, 2011.

[TTOH01] Yasuyuki Tahara, Nobukazu Toshiba, Akihiko Ohsuga, Shinichi Honiden. Secure and efficient mobile agent application reuse using patterns. *Proceedings of the 2001 symposium on Software reusability: putting software reuse in context*, SSR '01, strony 78–85, New York, USA, 2001. ACM.

[Tur36] Alan M. Turing. On Computable Numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[Tur50] Alan M. Turing. Computing Machinery and Intelligence. *Mind*, LIX:433–460, 1950.

[WDK11] Jaclyn Wainer, Laura Dabbish, Robert Kraut. Should I open this email?: inbox-level cues, curiosity and attention to email. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, strony 3439–3448, New York, USA, 2011. ACM.

[Weg97] Peter Wegner. Why Interaction Is More Powerful Than Algorithms. *Communications of the ACM*, 40(5):80–91, 1997.

[WfM99] WfMC. Workflow Management Coalition. Terminology and Glossary. Raport techniczny WFMC-TC-1011, Issue 3.0, Workflow Management Coalition, Winchester, United Kingdom, 1999.

[WfM12a] WfMC. Workflow Management Coalition. XPDL Schema file. http://www.xpdl.org/standards/xpdl-2.2/bpmnxpdl_40a.xsd, 2012.

[WfM12b] WfMC. Workflow Management Coalition Workflow Standard. Process Definition Interface - XML Process Definition Language (Version 2.2). Raport techniczny WFMC-TC-1025, Workflow Management Coalition, 2012.

[Wor98] World Wide Web Consortium. W3C XML Specification DTD. http://www.w3.org/XML/1998/06/xmlspec-report-19980910.htm, 1998.

[Wor12] World Wide Web Consortium. W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures. http://www.w3.org/TR/xmlschema11-1/, 2012.

[WRH+09] Petia Wohed, Nick Russell, Arthur H. M. ter Hofstede, Birger Andersson, Wil M. P. van der Aalst. Patterns-based evaluation of open source BPM systems: The cases of jBPM, OpenWFE, and Enhydra Shark. *Inf. Softw. Technol.*, 51(8):1187–1216, 2009.

[WS96] Steve Whittaker, Candace Sidner. Email overload: exploring personal information management of email. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '96, strony 276–283, New York, USA, 1996. ACM.

[Yao11] Yaoqiang. Yaoqiang XPDL Editor. http://xpdl.yaoqiang.org, 2011.

[Zyp11] Kris Zyp. A JSON Media Type for Describing the Structure and Meaning of JSON Documents. http://tools.ietf.org/html/draft-zyp-json-schema-03, 2011.

Spis rysunków

Załącznik A

Schematy XML Schema definiujące komponenty dokumentu MIND

A.1. Schemat komponentu głównego: document.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="mime-types.xsd"/>
  <xs:include schemaLocation="common-types.xsd"/>
  <xs:element name="document" type="documentType"/>
  <xs:complexType name="documentType">
    <xs:sequence>
      <xs:element name="head" type="headType"/>
    </xs:sequence>
    <xs:attribute name="id" type="hubdocIdAttributeType"
    use="required"/>
    <xs:attribute name="security" type="securityType" default="none"/>
    <xs:attribute name="extensible" type="yesNoType" default="yes"/>
  </xs:complexType>
  <xs:complexType name="headType">
    <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="meta" type="metaType" minOccurs="0"
      maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
```

```
  <xs:complexType name="metaType">
    <xs:attribute name="name" type="metaNameType"/>
    <xs:attribute name="content" type="xs:normalizedString"/>
  </xs:complexType>
  <xs:simpleType name="metaNameType">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="Keywords"/>
      <xs:enumeration value="Version"/>
      <xs:enumeration value="Description"/>
      <xs:enumeration value="Author"/>
      <xs:enumeration value="Generator"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

A.2. Schemat komponentu usług: services.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="common-types.xsd"/>
  <xs:element name="services" type="servicesType"/>
  <xs:complexType name="servicesType">
    <xs:sequence>
      <xs:element name="embeded-service" type="serviceEmbededType"
      minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="local-service" type="serviceLocalType"
      minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="external-service" type="serviceExternalType"
      minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="script-path" type="xs:normalizedString"/>
  </xs:complexType>
  <xs:complexType name="serviceEmbededType">
    <xs:attribute name="id" type="serviceIdAttributeType"
    use="required"/>
    <xs:attribute name="name" type="xs:normalizedString"
    use="required"/>
    <xs:attribute name="description" type="xs:string"/>
  </xs:complexType>
```

```
  <xs:complexType name="serviceLocalType">
    <xs:attribute name="id" type="serviceIdAttributeType"
    use="required"/>
    <xs:attribute name="name" type="xs:normalizedString"
    use="required"/>
    <xs:attribute name="description" type="xs:string"/>
    <xs:attribute name="application" type="xs:normalizedString"
    use="required"/>
  </xs:complexType>
  <xs:complexType name="serviceExternalType">
    <xs:attribute name="id" type="serviceIdAttributeType"
    use="required"/>
    <xs:attribute name="name" type="xs:normalizedString"
    use="required"/>
    <xs:attribute name="description" type="xs:string"/>
    <xs:attribute name="host" type="xs:anyURI" use="required"/>
  </xs:complexType>
</xs:schema>
```

A.3. Schemat komponentu pracowników wiedzy: workers.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
       <xs:include schemaLocation="common-types.xsd"/>
       <xs:include schemaLocation="worker-position-types.xsd"/>
       <xs:element name="workers" type="workersType"/>
       <xs:complexType name="workersType">
              <xs:sequence>
                     <xs:element name="worker" type="workerType"
                     maxOccurs="unbounded"/>
              </xs:sequence>
       </xs:complexType>
       <xs:complexType name="workerType">
              <xs:sequence>
                     <xs:element name="name"
                     type="xs:normalizedString"/>
                     <xs:element name="position"
                     type="workerPositionType"/>
```

```
                <xs:element name="description" type="xs:string"
                minOccurs="0"/>
                <xs:element name="contact" type="contactType"/>
                <xs:element name="communication"
                type="communicationType" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="id" type="workerIdAttributeType"
        use="required"/>
        <xs:attribute name="originator" type="yesNoType"/>
        <xs:attribute name="active" type="yesNoType"/>
</xs:complexType>
<xs:complexType name="contactType">
        <xs:sequence>
                <xs:element name="email-address"
                type="emailAddressType"/>
                <xs:element name="phone-number"
                type="phoneNumberType" minOccurs="0"/>
                <xs:element name="ip-address"
                type="ipAddressType" minOccurs="0"/>
                <xs:element name="webpage-url" type="xs:anyURI"
                minOccurs="0"/>
                <xs:element name="postal-address"
                type="xs:string" minOccurs="0"/>
        </xs:sequence>
</xs:complexType>
<xs:complexType name="communicationType">
        <xs:sequence>
                <xs:element name="simplex" type="workersListType"
                minOccurs="0"/>
                <xs:element name="duplex" type="workersListType"
                minOccurs="0"/>
                <xs:element name="broadcast"
                type="workersListType" minOccurs="0"/>
        </xs:sequence>
</xs:complexType>
<xs:simpleType name="workersListType">
        <xs:union memberTypes="workersIdListType">
                <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="ALL"/>
                        </xs:restriction>
```

```
                    </xs:simpleType>
                </xs:union>
        </xs:simpleType>
        <xs:simpleType name="workersIdListType">
                <xs:list itemType="workerIdAttributeType"/>
        </xs:simpleType>
</xs:schema>
```

A.4. Schemat komponentu dokumentów składowych: parts.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="authors.xsd"/>
  <xs:include schemaLocation="mime-types.xsd"/>
  <xs:include schemaLocation="common-types.xsd"/>
  <xs:element name="parts" type="partsType"/>
  <xs:complexType name="partsType">
    <xs:sequence>
      <xs:element name="part" type="partType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="partType">
    <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="description" type="xs:string" minOccurs="0"/>
      <xs:element name="modified" type="xs:dateTime"/>
      <xs:element name="content">
        <xs:complexType>
          <xs:attribute name="type" type="mimeType" use="required"/>
          <xs:attribute name="href" type="xs:anyURI"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="access" type="accessType" minOccurs="0"/>
      <xs:element name="annotations" type="annotationsType"
      minOccurs="0"/>
      <xs:element name="attachments" type="attachmentsType"
      minOccurs="0"/>
    </xs:sequence>
```

```
  <xs:attribute name="id" type="partIdAttributeType" use="required"/>
  <xs:attribute name="state" type="partStateType"/>
</xs:complexType>
<xs:complexType name="partContentType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="type" type="mimeType"/>
      <xs:attribute name="encoding" type="encodingType"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:complexType name="accessType">
  <xs:sequence>
    <xs:element name="none" type="workersListType" minOccurs="0"/>
    <xs:element name="toc" type="workersListType" minOccurs="0"/>
    <xs:element name="read" type="workersListType" minOccurs="0"/>
    <xs:element name="print" type="workersListType" minOccurs="0"/>
    <xs:element name="annotate" type="workersListType" minOccurs="0"/>
    <xs:element name="write" type="workersListType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="annotationsType">
  <xs:sequence>
    <xs:element name="annotation" type="annotationType"
    maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="annotationType">
  <xs:sequence>
    <xs:element name="text" type="annotationsTextType" minOccurs="0"
    maxOccurs="unbounded"/>
    <xs:element name="image" type="annotationsImageType"
    minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="annotationIdAttributeType"
  use="required"/>
  <xs:attribute name="name" type="xs:NMTOKEN"/>
  <xs:attribute name="worker" type="workerIdAttributeType"/>
  <xs:attribute name="annotation" type="annotationIdAttributeType"/>
</xs:complexType>
<xs:complexType name="annotationsTextType">
```

```
    <xs:sequence>
      <xs:element name="content" type="annotationContentType"
      maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="start" type="xs:nonNegativeInteger"/>
    <xs:attribute name="end" type="xs:nonNegativeInteger"/>
  </xs:complexType>
  <xs:complexType name="annotationsImageType">
    <xs:sequence>
      <xs:element name="content" type="annotationContentType"
      maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="src" type="xs:normalizedString"/>
    <xs:attribute name="x" type="xs:nonNegativeInteger"/>
    <xs:attribute name="y" type="xs:nonNegativeInteger"/>
    <xs:attribute name="width" type="xs:nonNegativeInteger"/>
    <xs:attribute name="height" type="xs:nonNegativeInteger"/>
  </xs:complexType>
  <xs:complexType name="annotationContentType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="type" type="mimeType" use="required"/>
        <xs:attribute name="href" type="xs:anyURI"/>
        <xs:attribute name="part" type="partIdAttributeType"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="attachmentsType">
    <xs:sequence>
      <xs:element name="attachment" type="attachmentType"
      maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="attachmentType">
    <xs:sequence>
      <xs:element name="content" type="attachmentContentType"/>
    </xs:sequence>
    <xs:attribute name="id" type="attachmentIdAttributeType"
    use="required"/>
    <xs:attribute name="name" type="xs:NMTOKEN"/>
    <xs:attribute name="worker" type="workerIdAttributeType"/>
```

```
    <xs:attribute name="annotation" type="annotationIdAttributeType"/>
  </xs:complexType>
  <xs:complexType name="attachmentContentType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="type" type="mimeType" use="required"/>
        <xs:attribute name="href" type="xs:anyURI"/>
        <xs:attribute name="part" type="partIdAttributeType"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:schema>
```

A.5. Schemat pomocniczych typów danych: common-types.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- id attribute types: -->
  <xs:simpleType name="hubdocIdAttributeType">
        <xs:restriction base="xs:ID">
          <xs:pattern value="MEN\d{5,}"/>
        </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="templateIdAttributeType">
        <xs:restriction base="xs:ID">
          <xs:pattern value="TMP\d{5,}"/>
        </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="serviceIdAttributeType">
        <xs:restriction base="xs:NMTOKEN">
          <xs:pattern value="SRV\d{5,}"/>
        </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="workerIdAttributeType">
        <xs:restriction base="xs:NMTOKEN">
          <xs:pattern value="AUT\d{5,}"/>
        </xs:restriction>
  </xs:simpleType>
```

```
<xs:simpleType name="partIdAttributeType">
      <xs:restriction base="xs:NMTOKEN">
        <xs:pattern value="PAR\d{5,}"/>
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="annotationIdAttributeType">
      <xs:restriction base="xs:NMTOKEN">
        <xs:pattern value="ANN\d{5,}"/>
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="attachmentIdAttributeType">
      <xs:restriction base="xs:ID">
        <xs:pattern value="ATT\d{5,}"/>
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="nodeIdAttributeType">
      <xs:restriction base="xs:ID">
        <xs:pattern value="NOD\d{5,}"/>
      </xs:restriction>
</xs:simpleType>
<!-- common types: -->
<xs:simpleType name="partStateType">
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="new"/>
        <xs:enumeration value="modified"/>
        <xs:enumeration value="verified-positive"/>
        <xs:enumeration value="verified-negative"/>
        <xs:enumeration value="done"/>
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="securityType">
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="none"/>
        <xs:enumeration value="low"/>
        <xs:enumeration value="medium"/>
        <xs:enumeration value="high"/>
        <xs:enumeration value="paranoid"/>
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="phoneNumberType">
      <xs:restriction base="xs:normalizedString">
```

```
        <xs:pattern value="\+?([0-9]+\s?)+"/>
      </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="emailAddressType">
      <xs:restriction base="xs:string">
        <xs:pattern value="[\w\.]+@[\w\.]+"/>
      </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ipAddressType">
      <xs:restriction base="xs:string">
        <xs:pattern value="([0-9]{1,3}\.){3}[0-9]{1,3}"/>
      </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="yesNoType">
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="yes"/>
        <xs:enumeration value="no"/>
        <xs:enumeration value="true"/>
        <xs:enumeration value="false"/>
      </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="encodingType">
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="ISO-8859-2"/>
        <xs:enumeration value="UTF-8"/>
      </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Załącznik B

Diagramy wzorców koordynacji dokumentów utworzone w TWE

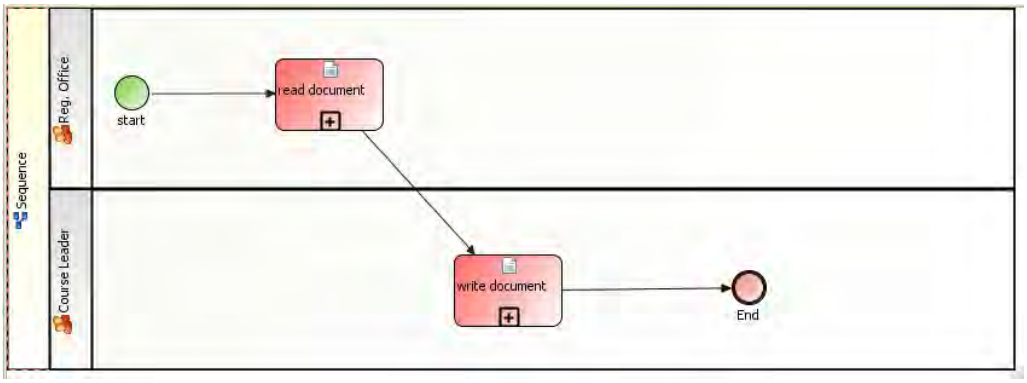
Diagramy przedstawione w tym załączniku, zostały utworzone w programie *Together Workflow Editor (TWE)* oraz zrealizowane w aplikacji zarządzającej przepływem pracy *Together Workflow Server (TWS)*. Zamieszczone rysunki są bezpośrednimi zrzutami ekranu wykonanymi dla TWE. TWE nie zawsze poprawnie przedstawia tranzycje warunkowe, oznaczając je jako bezwarunkowe (por. rys. 5.3, str. 103).

Diagramy te zawierają przykładowe realizacje wzorców koordynacji dokumentów (podrozdział 5.3) dla studium przypadku zarządzania przedmiotem dydaktycznym.

B.1. Wzorce z rozproszonym stanem dokumentu

Sekwencyjny przepływ dokumentu

Pierwszy ze zrealizowanych wzorców (rys. B.1) jest uproszczoną sekwencją pole-

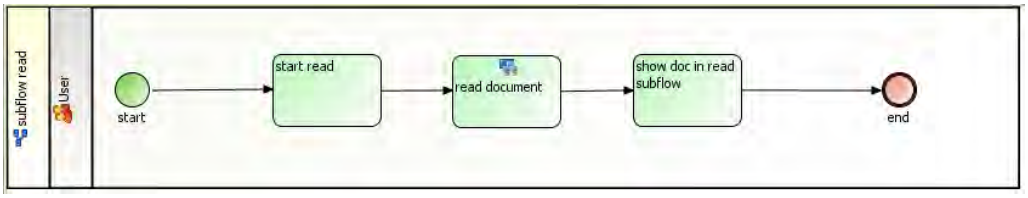


Rys. B.1. Sekwencja uproszczona

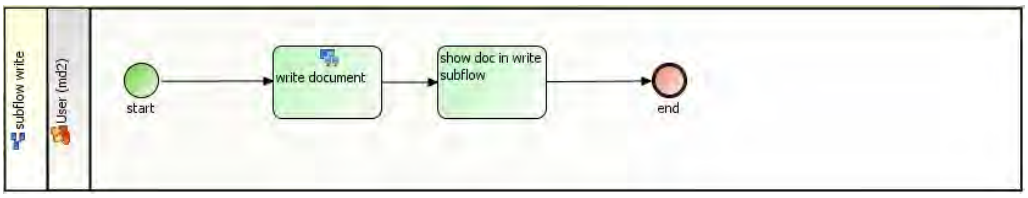
gającą na odczytaniu dokumentu z katalogu pracownika `Reg. Office` (pracownik

dziekanatu) i zapisaniu go w katalogu pracownika `Course Leader` (kierownik przedmiotu). Dokument jest przekazywany w całości i nie jest wykonywana żadna transformacja.

Wzorzec składa się z dwóch podprocesów: odczytującego dokument z dysku i przypisującego go zmiennej procesu (`subflow read`, rys. B.2) oraz zapisującego wartość zmiennej procesu do pliku XML (`subflow write`, rys. B.3). Czyn-



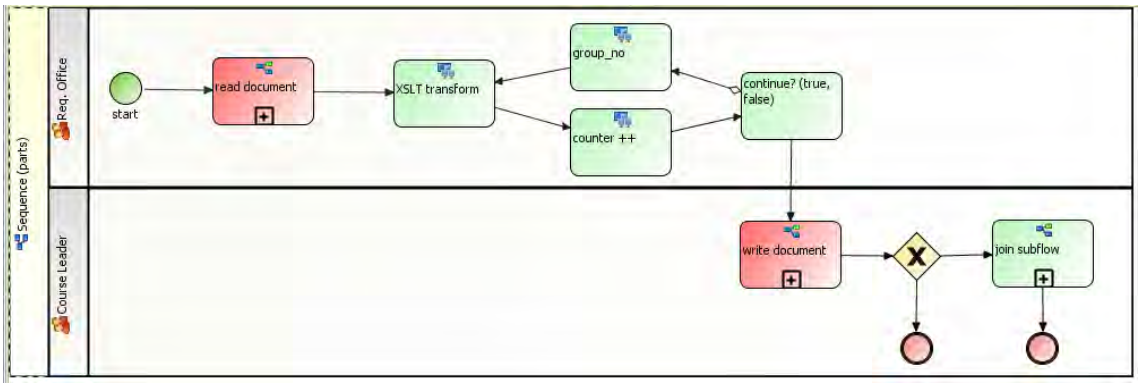
Rys. B.2. Podproces czytający dokument z dysku



Rys. B.3. Podproces zapisujący dokument na dysku

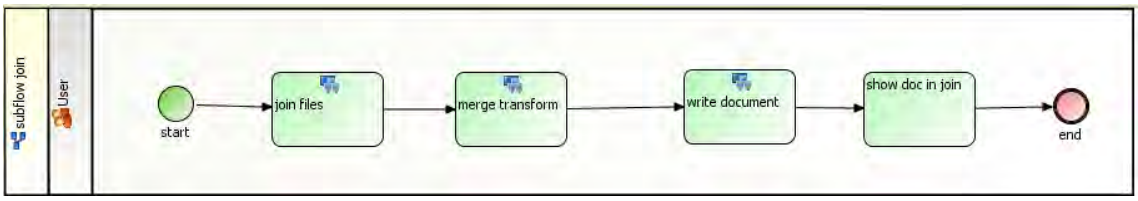
ności aplikacyjne `read document` i `write document` wykonywane są w TWS automatycznie, więc aby móc śledzić ich wykonywanie, umieściłam je w podprocesach. Czynności `show doc in read/write subflow` zawierają dodatkowy atrybut `VariableToProcess_VIEW` zdefiniowany w TWS do wyświetlania wartości zmiennych. Dzięki temu można obserwować, czy zawartość dokumentu została wczytana do zmiennej procesu.

Rysunek B.4 przedstawia ogólny wzorzec sekwencyjny, który pozwala przekazać



Rys. B.4. Sekwencja

kolejnemu pracownikowi dokument za jednym razem w całości lub iteracyjne w częściach, np. lista każdej grupy studenckiej jest przesyłana osobno. Podział następuje w czynności `XSLT transform` i wydzielanie kolejnych grup trwa, dopóki wartość zmiennej `continue` nie zostanie zmieniona na `false`. Po stronie odbiorcy dokumenty zapisywane są na dysku, zaś po zmianie wartości zmiennej `continue` na `false` zostaje uruchomiony podproces scalający otrzymane dokumenty `join subflow`, przedstawiony na rysunku B.5.



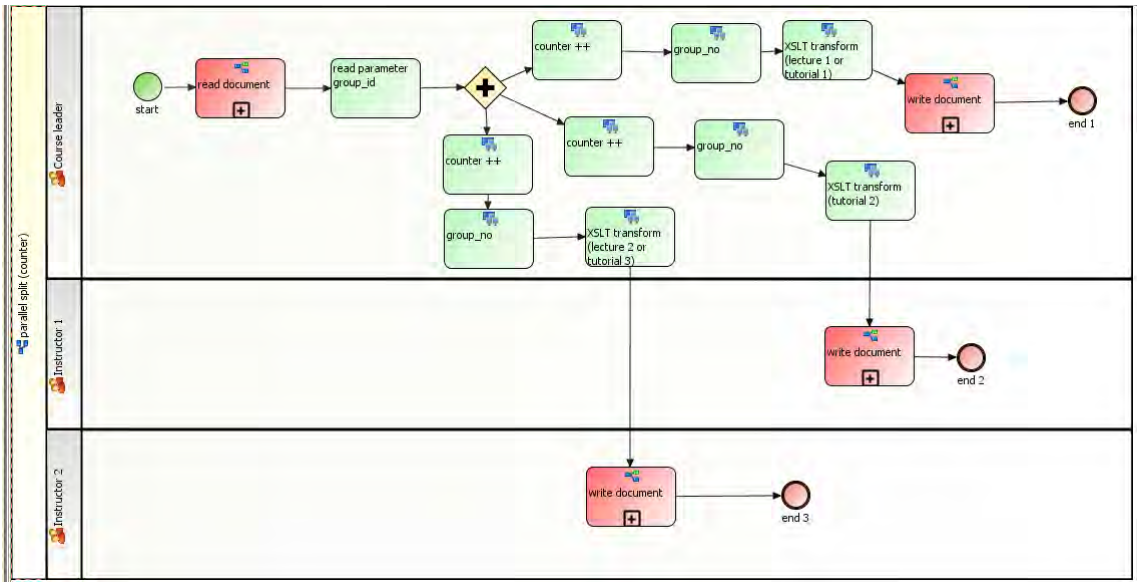
Rys. B.5. Podproces scalający dokument

Podproces scalający dokument ma zdefiniowane następujące czynności: `join files` łączy dokumenty XML w jeden plik o elemencie nadrzędnym `<join>`, `merge transform` wykonuje transformację uzyskanego pliku do postaci listy studentów, czyli scala treść dokumentów, `write document` zapisuje scalony dokument na dysku i `show doc in join` kontrolnie wyświetla wartość zmiennej, przechowującej treść scalonego dokumentu.

Wzorzec rozdzielający dokument

Wzorzec rozdzielający dokument, przedstawiony na rysunku B.6, po wczytaniu dokumentu z dysku, pozwala na wybór sposobu jego transformacji. W ramach czynności `read parameter group_id` użytkownik podaje wartość parametru `group_id`, względem którego nastąpi podział dokumentu na części. Transformacja rozdzielająca może przyjmować wartości parametru: `lecture, tutorial, seminar, project, lab, all, no`, zaś w podanym przykładzie na rysunku B.6 przepływ jest realizowany dla jednej z dwóch wartości parametru `lecture` i `tutorial`, co może być jednak dowolnie zmieniane. Użycie parametru `all` powoduje skopiowanie całej zawartości dokumentu.

Tranzycje wychodzące z czynności „bramki", są czynnościami warunkowymi, a warunki są spełnione w zależności od wartości paramentu `group_id`. W prezentowanym przykładzie, dla wartości parametru `tutorial`, wątek przechodzi przez wszystkie tranzycje, zaś dla wartości `lecture`, wybierana jest pierwsza i trzecie tranzycja. Następnie dla każdej wybranej ścieżki, realizowane są następujące czynności: `counter++` zwiększa wartość zmiennej `counter` o jeden, `group_no` przypisuje wartość zmiennej, pozwalającej na wybór numeru grupy zajęciowej w zależności od



Rys. B.6. Rozdzielenie dokumentu

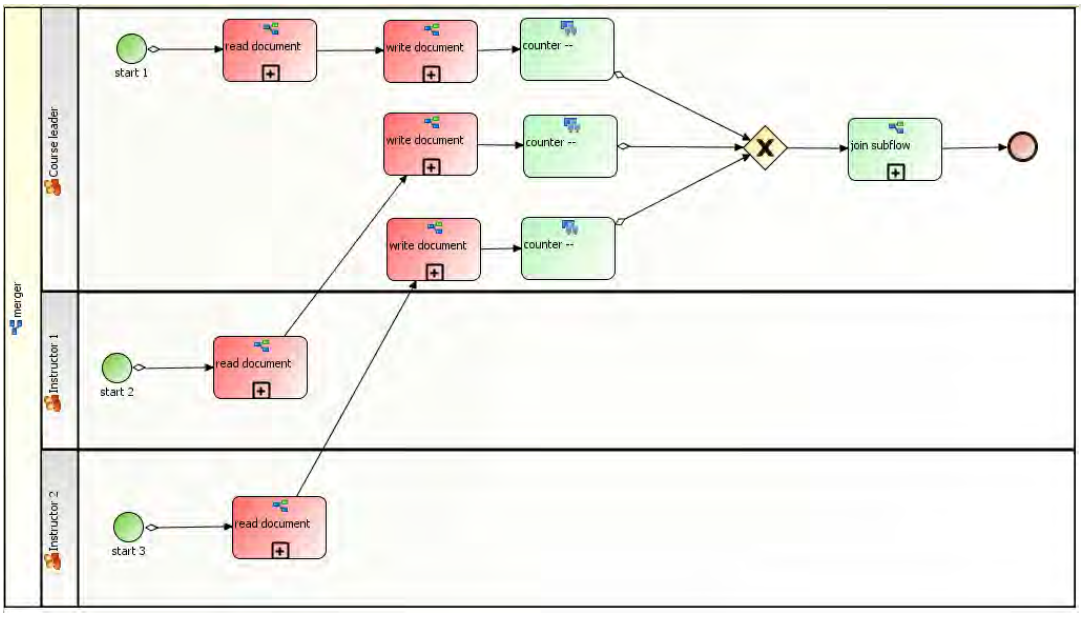
rodzaju zajęć, `XSLT transform` uruchamia transformację rozdzielającą dokument na grupy, `write document` zapisuje dokument w strukturze katalogów wskazanego przez proces pracownika.

Wzorce łączące dokument

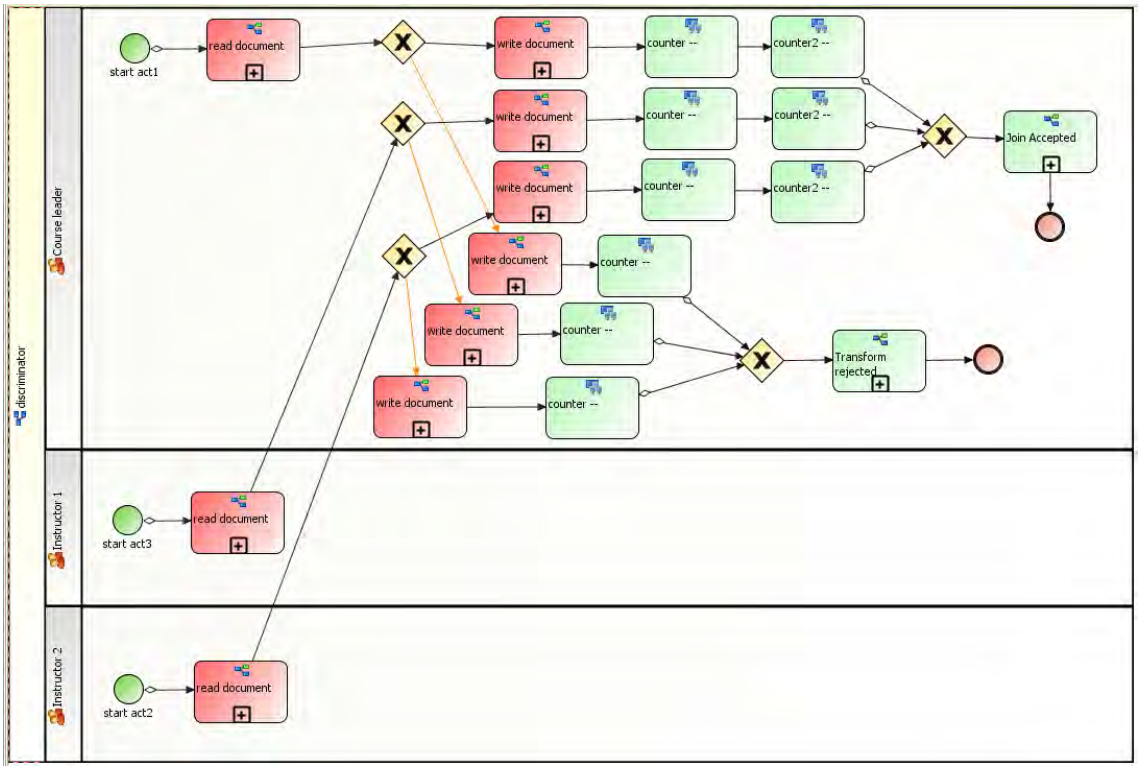
Wzorce łączące dokument (rys. B.7 i B.8) są komplementarne względem wzorca rozdzielającego. Dlatego mają element `<FormalParameters>`, specyfikujący formalne parametry, niezbędne do kontynuowania procesu po rozdzieleniu dokumentów. Są to `counter`, określający, ile dokumentów składowych powstało w puncie rozdzielenia oraz `group_id`, czyli wartość parametru transformacji rozdzielającej. Formalne parametry są wczytywane przez TWS jako parametry wejściowe procesu.

Na początku realizacji wzorców łączących znajdują się tranzycje warunkowe, pozwalające na wybór ścieżki zgodnie z wartością atrybutu `group_id` ustalonego we wzorcu rozdzielającym. Jest to potrzebne w prezentowanych przykładach, gdyż były testowane jako odrębne procesy, chociaż w praktyce wzorce te są kontynuacją procesu, w którym wcześniej nastąpiło rozdzielenie dokumentów. Zasadniczo, w punkcie łączenia potrzebna jest informacja o liczbie oczekiwanych dokumentów, czyli wartość zmiennej `counter`.

Wartość zmiennej `counter` jest zmniejszana o jeden z każdym wpłynięciem dokumentu do punktu łączenia. Wzorzec dyskryminującego łączenia dokumentów (rys. B.8) zawiera dodatkowo zmienną `counter2`, która określa, ile dokumentów ma być zaakceptowanych do dalszego przetwarzania – reszta zaś jest odrzucana. Wartość



Rys. B.7. Łączenie dokumentów



Rys. B.8. Dyskryminujące łączenie dokumentów

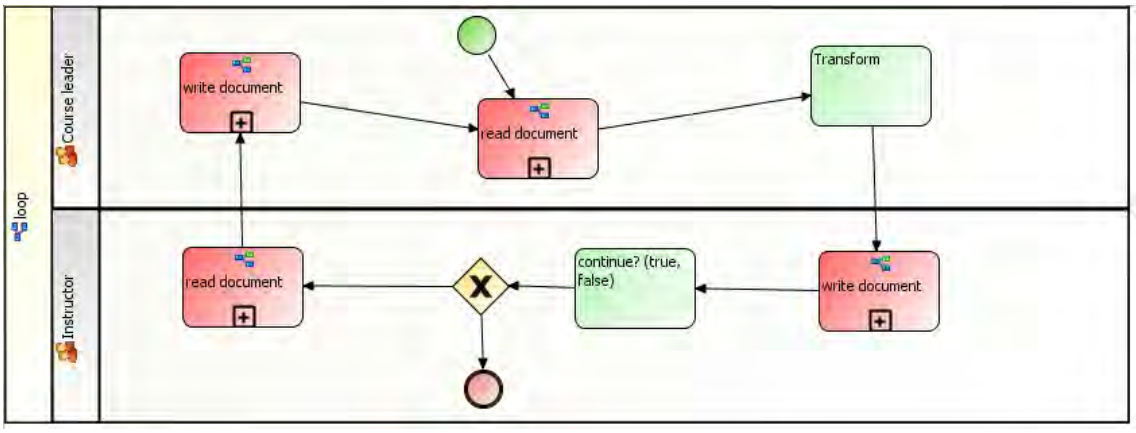
zmiennej `counter2` jest w danym przykładzie wpisana w pliku polityki, ale może być również podawana przez użytkownika.

Tranzycje wychodzące z czynności `counter--` posiadają warunek `counter==0`,

dzięki czemu czynność łącząca zostanie wykonana po wpłynięciu wszystkich oczekiwanych dokumentów. W dyskryminatorze, tranzycje wychodzące z czynności `counter2--` posiadają warunek `counter2==0`, a więc czynność `Join accepted` będzie wykonana po tym, jak wpłyną wszystkie akceptowane dokumenty. W tym przypadku proces może być kontynuowany, mimo że wzorzec będzie nadal oczekiwał na pozostałe dokumenty.

Wzorzec iteratora

Rysunek B.9 prezentuje przykład wzorca iterującego, który realizuje pętlę strukturalną z warunkiem sprawdzanym na wyjściu z pętli. Czynność `continue? (true, false)` posiada zdefiniowany dodatkowy atrybut `VariableToProcess_UPDATE`, dzięki któremu aplikacja TWS przy wykonywaniu tej czynności będzie umożliwiała użytkownikowi wprowadzenie wartości zmiennej decydującej o przebiegu pętli.



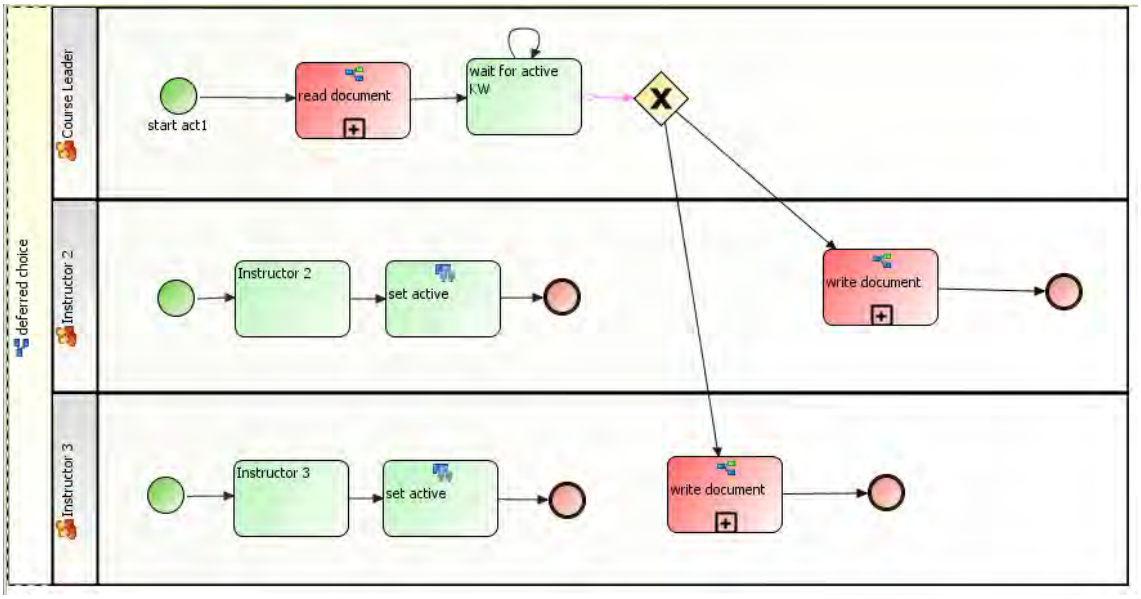
Rys. B.9. Iterator dokumentu

B.2. Wzorce ze sprzężonym stanem dokumentu

We wszystkich przykładach realizujących wzorce ze stanem sprzężonym dokumentu został wykorzystany element `<Deadline>`, umieszczony w czynnościach, które będę nazywać dalej czynnościami oczekującymi na sygnał. Przykładowa zawartość elementu `<Deadline>` jest przedstawiona na str. 141. Zmiana wartości zmiennej procesu w ramach wykonywania innej czynności (którą będę nazywać czynnością wysyłającą sygnał) powoduje ustawienie czasu realizacji czynności oczekującej na sygnał na 0 sekund, a więc powoduje jej automatyczne przerwanie i pobudzenie tranzycji typu `Exception`, która jest zaznaczona na diagramach kolorem różowym.

Wzorzec odroczonego wyboru

Wzorzec odroczonego wyboru zaprezentowany na rys. B.10 realizuje sytuację, w której dokument zostanie wysłany do pierwszego pracownika, który pojawi się w procesie (a dokładniej, zrealizuje czynność `Instructor_N`). Czynności `set active` są czynnościami wysyłającymi sygnał i zmieniają one wartość zmiennej procesu `activeKW` z zera na liczbę odpowiadającą numerowi instruktora (2 lub 3). Odpowiedni warunek sprawia, że modyfikacja wartości zmiennej `activeKW` następuje tylko wtedy, gdy wynosiła ona zero. Dzięki temu, niezależnie od chwili wykonania czynności odbierającej sygnał `wait for active KW`, dokument dotrze zawsze do pracownika, który zgłosił się jako pierwszy.

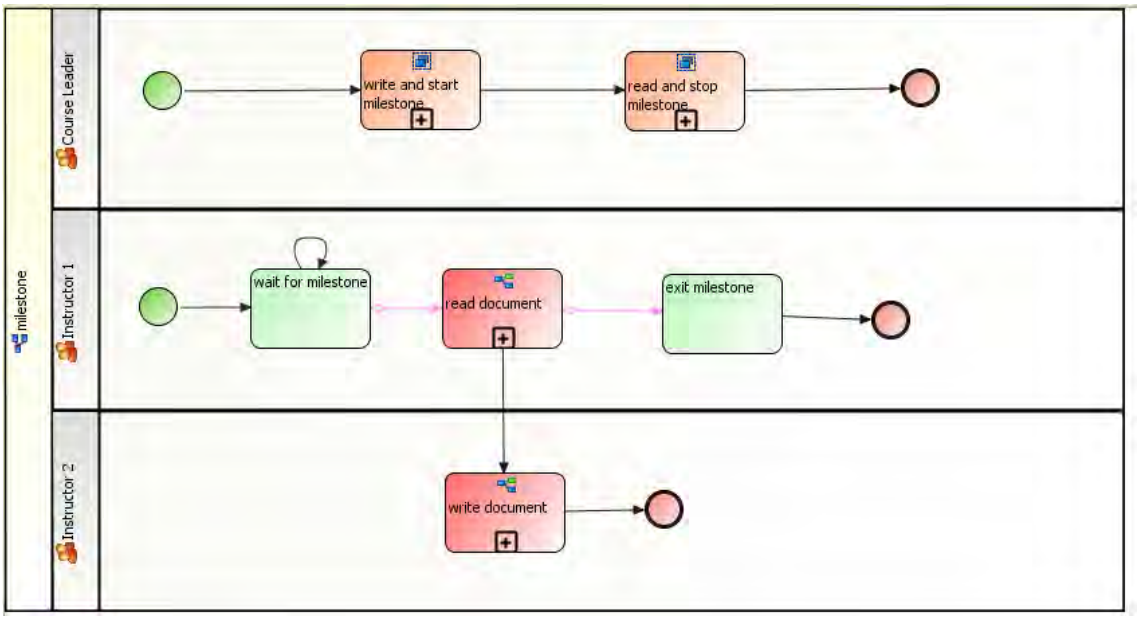


Rys. B.10. Odroczony wybór

Wzorzec kamienia milowego

Przykład wzorca kamienia milowego (rys. B.11) realizuje sytuację, w której pracownik `Instructor 2` może przekazać dokument pracownikowi `Instructor 3` tylko w czasie, gdy `Course leader` odebrał dokument, ale nie został on jeszcze wczytany jako zmienna do TWS. Podproces `write and start milestone` zapisuje dokument na dysku i wysyła sygnał rozpoczynający „kamień milowy". Wówczas przerywane jest wykonywanie czynności odbierającej sygnał `wait for milestone` i `Instructor 2` może wykonać czynność `read document`. Podproces `read and stop milestone` w chwili uruchomienia wysyła wpierw sygnał kończący „kamień milowy", a następnie wczytuje dokument do zmiennej TWS. Sygnał kończący

jest odbierany przez czynność `read document` i przerywa jej wykonywanie, o ile pracownik `Instructor 2` nie zdążył już jej zrealizować i przekazać dokumentu kolejnemu pracownikowi, zgodnie z jego ścieżką przepływu.



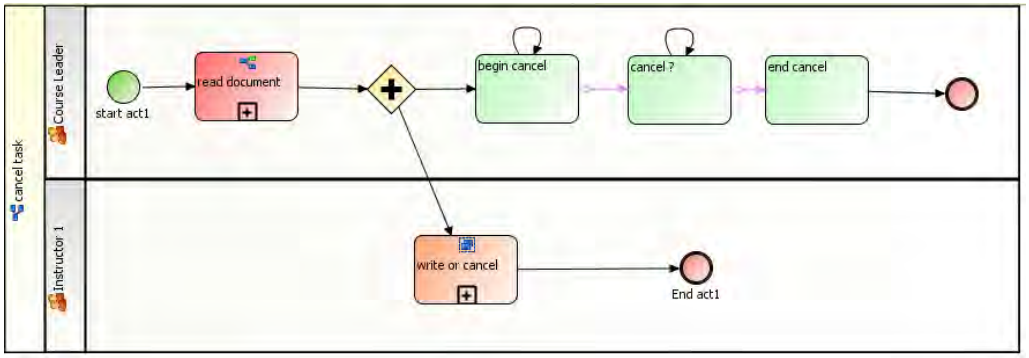
Rys. B.11. Kamień milowy

Wzorce anulujące

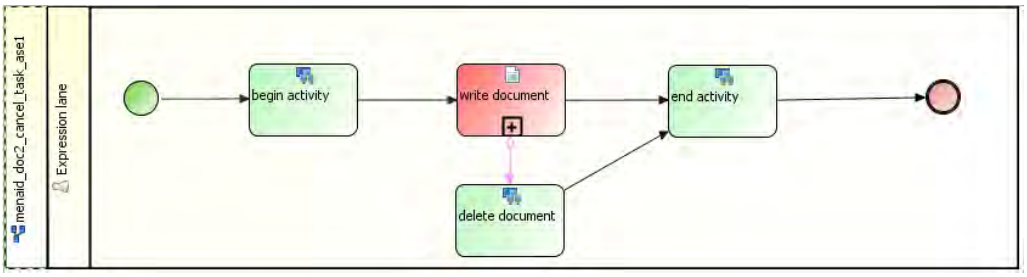
Rysunek B.12 przedstawia przykład anulowania czynności, przy czym ta możliwość jest dostępna tylko w czasie, gdy czynność, która może zostać anulowana, jest aktywna. Czynność, która może być anulowana, jest przedstawiona na rysunku B.12 jako podproces `write or cancel`, który pozwala zapisać dokument na dysku i usunąć go w przypadku odebrania sygnału anulującego. Graf tego podprocesu jest przedstawiony na rysunku B.13.

Czynność `begin cancel` oczekuje na sygnał od czynności rozpoczynającej podproces `begin activity`. Po jego otrzymaniu, pracownik `Course leader` może anulować czynność pracownika `Instructor 1`. Jeśli się na to zdecyduje, czynność podprocesu `write document` zostaje przerwana i wykonuje się czynność `delete document`, usuwająca zapisany dokument. Czynność `end activity` wysyła sygnał przerywający wykonywanie czynności `cancel?`.

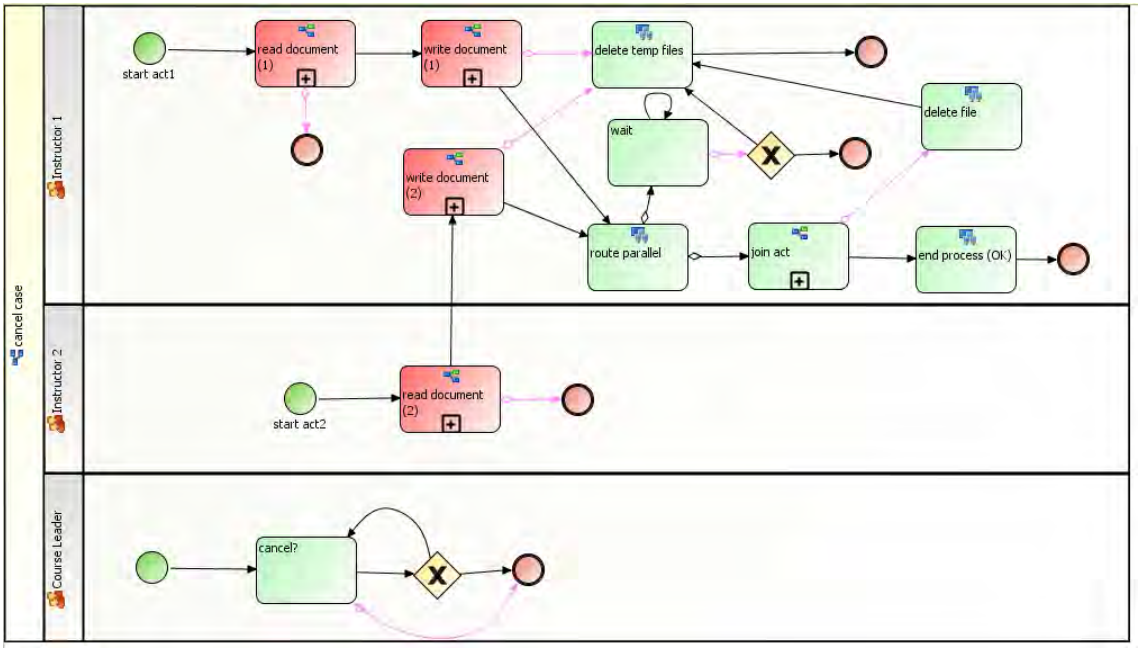
Przy anulowaniu czynności nie jest konieczne sprawdzanie, czy jest aktywna. Czynność może po prostu posiadać możliwość obierania sygnału anulującego i być przerywana w trakcie wykonywania lub przed rozpoczęciem działania. Innym problemem jest, co zrobić z dokonanymi już zmianami w dokumentach składowych.



Rys. B.12. Anulowanie czynności



Rys. B.13. Podproces anulowanej czynność



Rys. B.14. Anulowanie całego procesu

Rysunek B.14 przedstawia proces, w którym pracownik `Course leader` może anulować wszystkie czynności wykonywane przez instruktorów. Różowe strzałki prowadzą do czynności, które zostają wykonane po wysłaniu sygnału anulującego

i różnią się w zależności od tego, która czynność odbierze ów sygnał. Możliwość anulowania procesu kończy się w chwili realizacji czynności `end process (OK)`, która kończy proces z sukcesem.

Załącznik C

Proces zarządzania przedmiotem szkoły wyższej

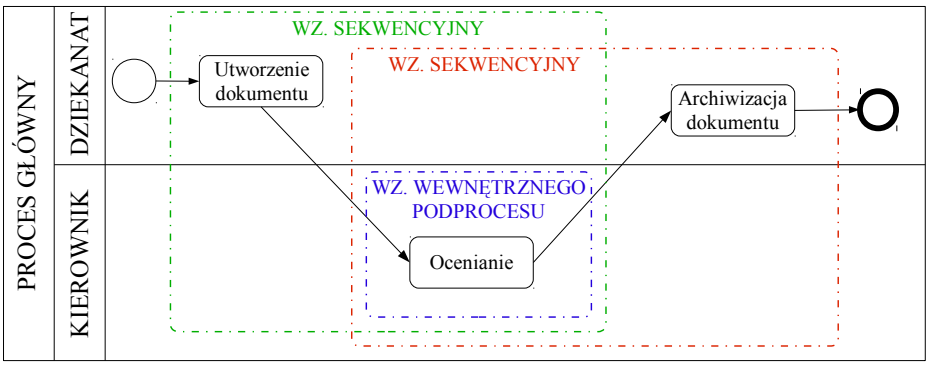
Proces zarządzania przedmiotem jest przykładem procesu wiedzy. Jego głównym celem jest formułowanie ocen studentów i dostarczenie gotowych list z ocenami do dziekanatu. Jak w modelu przedstawionym w rozdziale 2, wykorzystuje on trzy źródła wiedzy: jawną (skodyfikowaną) w postaci zbioru dokumentów (dokumentacji przedmiotu), obejmującego odebrane i ocenione prace studentów, listy obecności, kopie zaświadczeń lekarskich, notatki prowadzącego itp., niejawną (ugruntowaną), zawartą w procedurach związanych z realizowaniem przedmiotów na uczelni, opisanych ogólnie w regulaminie studiów i szczegółowo dla danego przedmiotu (np. harmonogram zaliczeń w semestrze, podawany na pierwszym wykładzie przez kierownika przedmiotu) oraz ukrytą (spersonalizowaną), wynikającą z doświadczenia zawodowego wszystkich nauczycieli zaangażowanych w prowadzenie jego aktualnej edycji. Nauczyciele, korzystając z wymienionych źródeł wiedzy, wystawiają oceny cząstkowe (formujące), na podstawie których podejmują decyzję o zaliczeniu przedmiotu i ocenie końcowej.

Jest to proces niealgorytmiczny, gdyż zarówno wystawianie oceny, jak i przebieg procesu, wymagają podejmowania decyzji przez człowieka. Wiele z tych decyzji może być sformalizowanych i podejmowanych automatycznie, np. wyznaczenie oceny końcowej na podstawie ustalonych w procedurze zaliczeń progów punktowych, jednak w procesie zarządzania przedmiotem występują również decyzje wymagające udziału człowieka, np. ocena samodzielności wykonania zadania projektowego, czy stopień zaangażowania członków zespołu studenckiego w przygotowanie i przeprowadzenie seminarium. Ponadto ze względu na brak kontroli nad zewnętrznym otoczeniem przez prowadzących i studentów, opisany wyżej proces wiedzy musi radzić sobie z każdym możliwym zdarzeniem, jakie mogłoby zajść podczas jego realizacji. Ze względu na ich mnogość i nieprzewidywalność, trudno jest projektantowi dokumentu, przy

tworzeniu struktury procesu, przewidzieć z góry wszystkie sytuacje, które mogą się zdarzyć podczas jego realizacji. Niezbędna staje się zatem możliwość dynamicznej modyfikacji ścieżki migracji dokumentu już podczas jej realizacji, poprzez dodawanie podprocesów określających czynności, których wymaga konkretna sytuacja. Decyzję o modyfikacji ścieżki procesu podejmują na bieżąco uprawnieni uczestnicy procesu, interpretując według swojej wiedzy obowiązujące procedury, pozostając w zgodzie nie tylko ze swoim doświadczeniem (wiedzą ukrytą), ale także ze zdrowym rozsądkiem, etyką zawodową, sumieniem i tym podobnymi „niealgorytmicznymi" aspektami podejmowania decyzji.

Diagramy C.1 – C.7 prezentują przykład procesu zarządzania przedmiotem, w którym wystąpiły różne sytuacje wymagające indywidualnych decyzji uczestników (nauczycieli). Zbiór tych sytuacji jest wystarczająco liczny, by zademonstrować wszystkie wzorce zdefiniowane przeze mnie w rozdziale 5. Dla czytelności, wzorce na diagramach zostały przedstawione w sposób uproszczony, zaś ich szczegółowy opis znajduje się w podrozdziale 5.3.

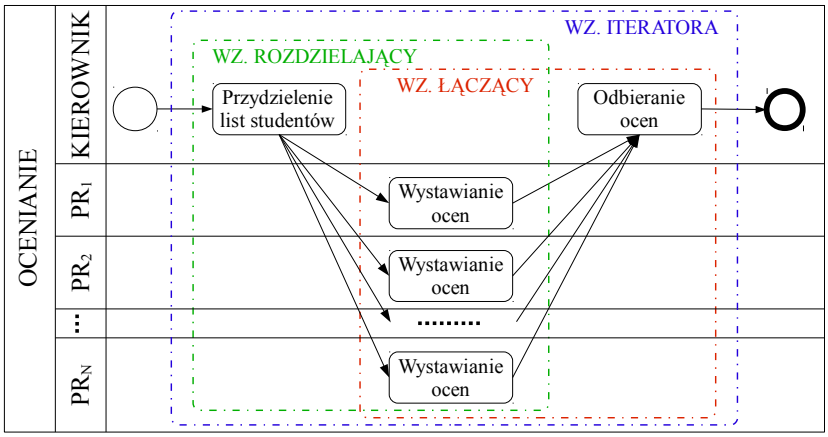
Rysunek C.1 prezentuje proces zarządzania przedmiotem na poziomie, na którym postrzega go dziekanat. Składa się on z wzorca sekwencyjnego przepływu listy



Rys. C.1. Proces główny

studentów z dziekanatu do kierownika przedmiotu, a po wykonaniu przez niego czynności oceniania, wypełniona lista w sposób sekwencyjny zostaje przekazana do dziekanatu. Kierownik jest odpowiedzialny za czynność oceniania i zna szczegóły jej realizacji, które nie są znane pracownikom dziekanatu. W ramach tej czynności dynamicznie definiuje wewnętrzny podproces, którego struktura jest zależna od rodzaju zajęć z danego przedmiotu (wykład, ćwiczenia, laboratoria itp.) i przypisanych im prowadzących.

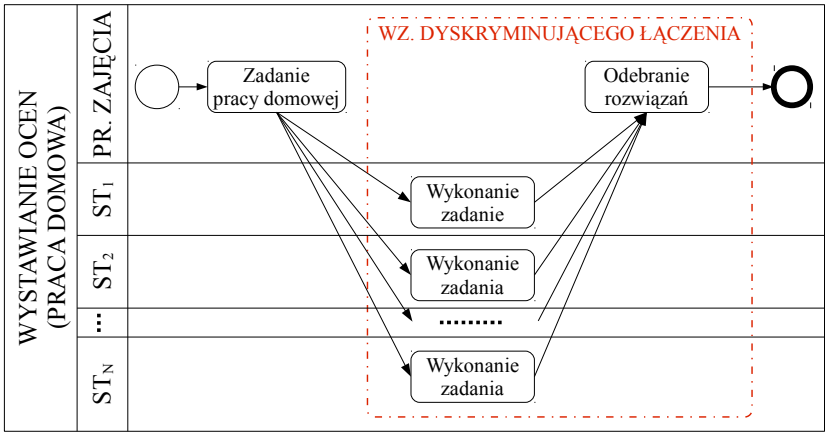
Wewnętrzny podproces oceniania jest przedstawiony na rysunku C.2. Kierownik przydziela listy studentów poszczególnych grup nauczycielom prowadzącym zajęcia (PR_1 – PR_N), całościowa lista studentów przypisanych do przedmiotu jest więc



Rys. C.2. Podproces oceniania

rozdzielana na składowe listy poszczególnych grup i dostarczana przypisanym nauczycielom (wzorzec rozdzielający). Do poszczególnych nauczycieli należy czynność wystawiania ocen cząstkowych i po tej czynności, listy składowe wracają do kierownika, u którego następuje ich scalenie (wzorzec łączący). Listy ocen mogą wracać do kierownika po wystawieniu każdej oceny cząstkowej, co odpowiada wzorcowi iteratora.

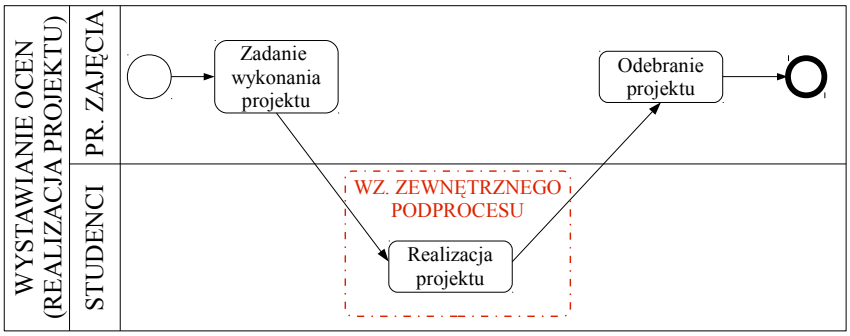
Wystawianie ocen może być prostą czynnością polegającą na wpisaniu odpowiednich wartości do dokumentu, ale mogą się zdarzyć sytuacje wymagające zdefiniowania podprocesu z użyciem różnych wzorców. Na przykład, diagram C.3 przedstawia sytuację oceniania pracy domowej, w której student, który dostarczy rozwiązanie jako pierwszy, otrzyma dodatkowe punkty, co odpowiada wzorcowi dyskryminującego łączenia.



Rys. C.3. Wystawianie ocen w ramach zajęć – ocenianie pracy domowej

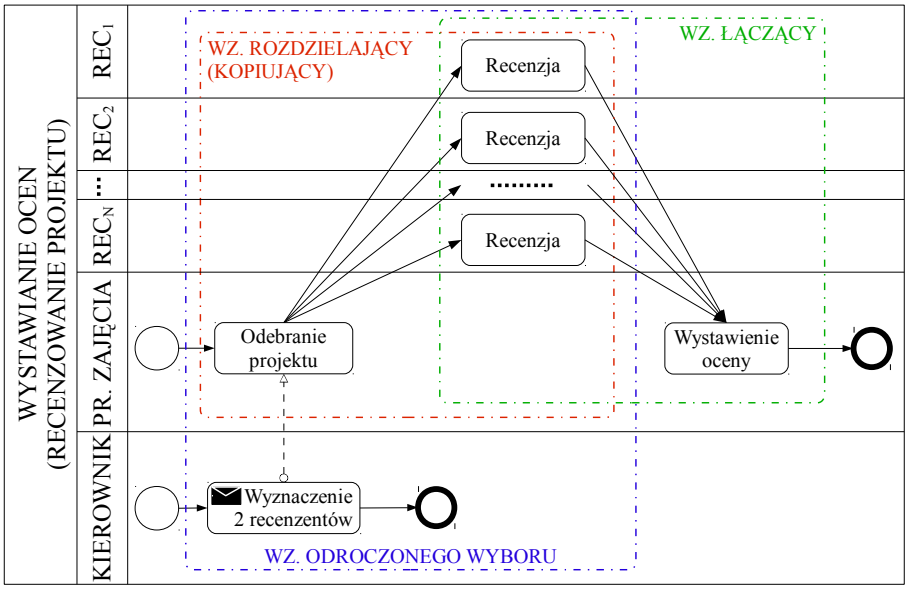
Rysunek C.4 prezentuje sytuację, która wystąpi, gdy studenci realizują projekt grupowy w zewnętrznej firmie; prowadzący projekt nie zna wówczas dokładnie

przebiegu procesu realizacji tego zadania projektowego, gdyż jest ono realizowane jako podproces zewnętrzny. Do prowadzącego należy odebranie zrealizowanego projektu lub jego poszczególnych etapów.



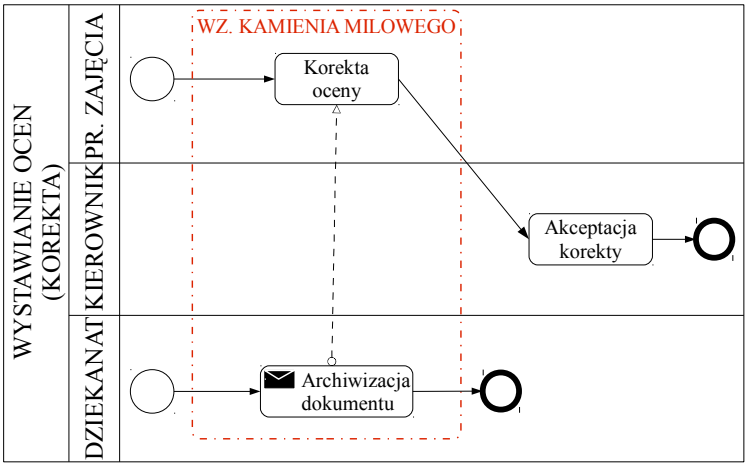
Rys. C.4. Wystawianie ocen w ramach zajęć – realizacja projektu w firmie zewnętrznej

Ocena wyniku realizacji zadania może wymagać opinii recenzentów, których arbitralnie wyznacza kierownik przedmiotu (wzorzec odroczonego wyboru), co prezentuje diagram C.5. Każdy ze wskazanych przez kierownika recenzentów otrzymuje od prowadzącego zajęcia kopię dokumentacji, na podstawie której opiniuje realizację projektu, co odpowiada wzorcowi rozdzielającemu w wersji kopiującej. Wzorzec łączący, w tym przypadku, będzie polegał na dostarczeniu każdej oczekiwanej recenzji prowadzącemu projekt, aby na ich podstawie mógł wystawić ocenę.



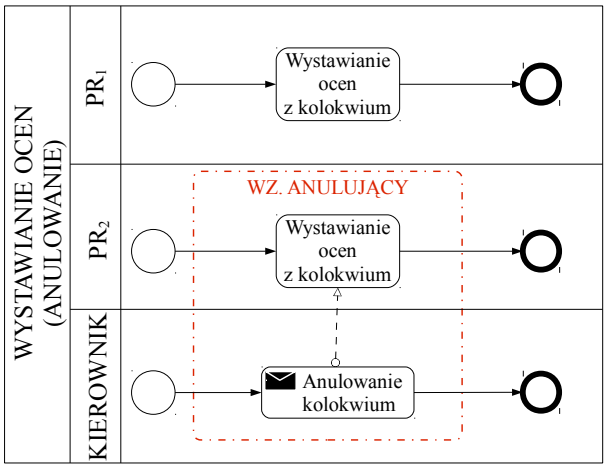
Rys. C.5. Wystawianie ocen w ramach zajęć – recenzowanie projektu

W procesie oceniania może wystąpić potrzeba dokonania korekty ocen, jak przedstawiono na rysunku C.6. Korekta jest jednak możliwa do czasu zatwierdzenia listy ocen w dziekanacie. Informacja z dziekanatu o zatwierdzeniu listy blokuje możliwość dokonywania w niej zmian przez prowadzących zajęcia, co odpowiada wzorcowi kamienia milowego.



Rys. C.6. Wystawianie ocen w ramach zajęć – korekta ocen

Kierownik przedmiotu ma również możliwość anulowania czynności wystawiania ocen (wzorzec anulujący) w sytuacji wykrycia pewnych nieprawidłowości w przebiegu oceniania, np. plagiatu zadania projektowego lub oszustwa (ściągania) na kolokwium. Sytuację anulowania wystawiania ocen z kolokwium w jednej z grup przedstawia rysunek C.7.



Rys. C.7. Wystawianie ocen w ramach zajęć – anulowanie czynności oceniania