OŚRODEK
PRZETWARZANIA
INFORMACJI
PAŃSTWOWY INSTYTUT BADAWCZY

OSF

Obsługa Strumieni Finansowania

redaktor naukowy
Adam Bochenek

Systemy informatyczne
wspierające naukę i szkolnictwo wyższe

OSF:
Obsługa Strumieni Finansowania

Redakcja naukowa:
Adam Bochenek

OŚRODEK
PRZETWARZANIA
INFORMACJI
PAŃSTWOWY INSTYTUT BADAWCZY

Systemy informatyczne wspierające naukę i szkolnictwo wyższe
OSF: Obsługa Strumieni Finansowania

Redakcja naukowa:
Adam Bochenek

Redakcja techniczna:
Krystyna Fetera

Autorzy:
Jacek Bieliński: rozdział 4
Adam Bochenek: rozdział 4
Urszula Sułek: rozdział 1, 2
Przemysław Zydroń: rozdział 3

Recenzent:
dr hab. Zenon A. Sosnowski, prof. Politechniki Białostockiej

Wydawca:
Ośrodek Przetwarzania Informacji – Państwowy Instytut Badawczy
al. Niepodległości 188b
00-608 Warszawa
e-mail: opi@opi.org.pl
www.opi.org.pl

© Copyright by Ośrodek Przetwarzania Informacji – Państwowy Instytut Badawczy
Warszawa 2021
Wszelkie prawa zastrzeżone

ISBN 978-83-63060-22-0

Projekt okładki i opracowanie graficzne:
Karina Maszewska

Skład i łamanie:
dr Jakub Kierzkowski

Druk:
Drukarnia DSS Szczepan Szymański
ul. Wolska 108, 05-119 Wola Aleksandra

Szanowni Państwo,

rozwój społeczno-gospodarczy każdego państwa w dużej mierze zależy od innowacyjności i zdolności do szybkiego wdrażania nowoczesnych rozwiązań. Nie byłoby to możliwe bez osiągnięć oraz wytrwałej pracy naukowców i badaczy z różnych dziedzin. Dlatego też w Ministerstwie Edukacji i Nauki podejmujemy wiele działań, które efektywnie zwiększają potencjał sektora badawczego. Przeznaczamy środki krajowe i europejskie, aby polska nauka była nowoczesna i innowacyjna. Warto również podkreślić, że rządowy program „Polski Ład" zawiera wiele strategicznych inwestycji. Będą one w jeszcze większym stopniu wspierać całe środowisko naukowe.

Olbrzymie nakłady finansowe, które rząd przeznacza na zwiększenie potencjału polskiej nauki, wymagają zastosowania bezpiecznego i nowoczesnego narzędzia informatycznego. W tym właśnie celu Ośrodek Przetwarzania Informacji – Państwowy Instytut Badawczy (OPI PIB) opracował Zintegrowany System Usług dla Nauki / Obsługa Strumieni Finansowania (ZSUN OSF). Pierwsza wersja tego systemu powstała już w 2005 r. i od tego czasu jest stale udoskonalana przez zespół Laboratorium Systemów Biznesowych OPI PIB. Dzięki temu nowoczesnemu narzędziu IT składanie wniosków o granty oraz proces uzyskiwania środków na działania badawczo-rozwojowe przebiega płynnie i transparentnie. Dodatkowo zespół ekspertów OPI PIB zapewnia wszystkim użytkownikom wsparcie merytoryczne i pomoc techniczną. W istotny sposób usprawnia to proces ubiegania się o dofinansowanie na działalność badawczo-naukową oraz przyspiesza całą procedurę.

Publikacja, którą Państwu przekazujemy, w kompleksowy i przystępny sposób przybliża rozwój ZSUN OSF. Znajdą w niej Państwo nie tylko rys historyczny związany z powstaniem systemu, ale także opis jego głównych funkcji. Autorzy w ciekawy sposób przedstawiają zalety narzędzia, a także opisują problemy związane z jego rozwojem. Poruszają kwestie dotyczące modernizacji i wprowadzania zmian, a jednocześnie podkreślają konieczność zagwarantowania stabilności narzędzia IT. Publikacja zawiera także wiele informacji na temat architektury systemu, które wcześniej nie były publikowane. Z pewnością każda osoba zainteresowana branżą IT oraz sektorem naukowo-badawczym znajdzie w monografii istotne dla siebie tematy i zagadnienia.

Serdecznie gratuluję ekspertom OPI PIB opracowania tej ciekawej i potrzebnej publikacji, a przede wszystkim serdecznie dziękuję za wdrożenie i stałą modernizację ZSUN OSF. Często docierają do mnie głosy ze środowiska naukowego, że narzędzie OPI PIB w istotny sposób usprawniło proces ubiegania się o dofinansowanie działalności naukowo-badawczej. ZSUN OSF podobnie jak POL-on, JSA czy ELA stał się uznaną „marką" instytutu, która jest gwarancją wysokiej jakości i bezpieczeństwa.

Serdecznie zachęcamy do lektury publikacji!

Wojciech Murdzek

Sekretarz Stanu,
Pełnomocnik Rządu do spraw reformy
funkcjonowania instytutów badawczych

Ministerstwo Edukacji i Nauki

SPIS TREŚCI

Rozdział 1. Koncepcja systemu **9**
Urszula Sułek

1.1. Tło historyczne ... 9
1.2. Główne cele i zadania systemu ... 10
1.3. Interesariusze ... 11

Rozdział 2. Procesy biznesowe **13**
Urszula Sułek

2.1. Wnioskowanie ... 14
2.2. Ocena wniosków ... 20
2.3. Obsługa finansowania ... 26
2.4. Rozliczanie finansowania ... 29

Rozdział 3. Proces technologiczny **31**
Przemysław Zydroń

3.1. Proces analizy wymagań ... 32
3.1.1. Analiza wstępna wymagań ... 32
3.1.2. Jeden punkt zgłoszeń ... 34
3.1.3. Analiza pełna etapu ... 34
3.2. Proces implementacji ... 35
3.2.1. Odesłane do poprawy ... 36
3.2.2. Odesłane do analityka ... 36
3.2.3. Uznane za wykonane ... 36
3.3. Proces testów ... 37
3.3.1. Testy release ... 39

Rozdział 4. Architektura systemu **41**
Adam Bochenek, Jacek Bieliński

4.1. Pierwsze decyzje architektoniczne ... 42
4.1.1. Dominujące na rynku technologie ... 42
4.1.2. Perspektywy technologii, trendy, wsparcie ... 43
4.1.3. Wybór technologii głównej ... 44

4.1.4. Pozostałe decyzje architektoniczne na etapie rozpoczynania projektu ... 45
4.2. Budowanie zespołu ... 47
4.2.1. Społeczeństwo informacyjne i klasa kreatywna ... 48
4.2.2. Zjawiska na rynku pracy w sektorze technologii informacyjnych i komunikacyjnych (ICT) ... 49
4.2.3. Kształcenie informatyków i sytuacja absolwentów informatyki na rynku pracy ... 51
4.2.4. Sytuacja absolwentów kierunków Informatyka na rynku pracy ... 55
4.2.5. Perspektywy ... 58
4.3. Rozwój i ewolucja systemu ... 59
4.3.1. Osiąganie równowagi pomiędzy koniecznością modernizacji a stabilnością systemu ... 59
4.4. Odejście od monolitu i nowa architektura ... 61
4.4.1. Architektura monolityczna ... 61
4.4.2. Problemy z monolitem ... 62
4.4.3. Mikrousługi ... 63
4.5. Złoty środek, nowa architektura OSF ... 65
4.5.1. Czy monolit ma wyłącznie wady ... 66
4.5.2. Modularny monolit ... 67
4.5.3. Warstwa utrwalania danych ... 67
4.5.4. DDD ... 68
4.5.5. UI ... 70

Rozdział 5. Zakończenie **71**

Spis ilustracji **73**

Bibliografia **75**

WSTĘP

Istnieje niewiele dziedzin, które rozwijają się tak dynamicznie jak informatyka. To nie tylko ciągłe doskonalenie istniejących produktów i technologii. Tutaj zmiany są tak fundamentalne, że możemy mówić o następujących co kilka lat kolejnych epokach. Czy ktoś z nas wyobrażał sobie dwadzieścia lat temu czasy, w których praktycznie każdy z nas będzie nosił w kieszeni urządzenie z kolorowym, dotykowym ekranem, wyposażone w ośmiordzeniowy procesor i gigabajty pamięci? Urządzenie, które połączone jest z całym światem, za pomocą którego można się komunikować, oglądać filmy, słuchać muzyki, płacić, nawigować? Z którym można się porozumieć w języku naturalnym?

Ten rozwój fascynuje i inspiruje. Ale to niewiarygodne tempo zmian ma różne oblicza. Na pewno nie zawsze idzie w parze z takimi cechami, jak niezawodność wynikająca ze stabilizacji czy nawet rutyny. A tam, gdzie pewność działania stawiamy na pierwszym miejscu są to przymioty bardzo pożądane.

Aplikacje, systemy komputerowe powinny być funkcjonalne. Powinny nam przede wszystkim służyć, realizować funkcje, które zaspokajają różne nasze potrzeby. Robić dobrze to, do czego zostały stworzone. Takie są nasze oczekiwania. Tak więc z jednej strony lubimy postęp, ale z drugiej nie zawsze pozytywnie postrzegamy zmiany, które następują zbyt szybko i niekoniecznie wiele wnoszą. Najlepszy jest, jak zwykle, złoty środek, kompromis. Jak to wygląda z perspektywy systemu informatycznego?

Mamy przyjemność zaprezentować Czytelnikom monografię poświęconą systemowi, który został stworzony 15 lat temu i od tego czasu działa nieprzerwanie. Działa nieustannie oferując nowe funkcje, jest cały czas rozwijany i modyfikowany. Ale ze względu na to, że musi być cały czas dostępny i niezawodny, nie zachodzą w nim rewolucyjne zmiany.

Ten system to OSF (Obsługa Strumieni Finansowania). Powstał w 2005 roku w Ośrodku Przetwarzania Informacji – Państwowym Instytucie Badawczym na zlecenie ówczesnego Ministerstwa Nauki i Informatyzacji. Głównym zadaniem twórców systemu w pierwszym okresie było stworzenie oprogramowania pozwalającego na jak najszybsze uruchomienie w trybie on-line naborów na konkursy związane z dofinansowaniem działalności badawczej. W kolejnych latach po uruchomieniu produkcyjnym systemu zespół wytwórczy skupiał się na rozwoju funkcjonalności oraz uruchamianiu kolejnych naborów. Aż w końcu, jak w przypadku każdej działalności produkcyjnej, nadszedł moment refleksji nad efektywnością procesu wytwórczego oraz stosowaną technologią informatyczną.

W niniejszej publikacji chcemy zapoznać Czytelnika z tym, co robi system i jak szeroki posiada obecnie wachlarz funkcjonalności, które notabene ciągle są rozwijane. Chcemy pokazać, z jakimi wyzwaniami mierzył się na przestrzeni lat produkcji i rozwoju systemu zespół wytwórczy – zarówno od strony zarządzania procesem produkcyjnym, jak i od strony zapewnienia optymalnej architektury, która odpowiadałaby potrzebom instytucji, na rzecz których system OSF funkcjonuje. Podstawowym pytaniem, jakie należałoby sobie zadać w takiej sytuacji jest to, czy po kilkunastu latach funkcjonowania produkcyjnego systemu stworzonego w konkretnej architekturze i technologii jakiekolwiek zmiany w tej materii są możliwe? Jeśli tak, to w jakim zakresie i w jakim trybie? Postaramy się w niniejszej pracy odpowiedzieć na te pytania.

Publikacja składa się z następujących rozdziałów:

1. Rozdział 1. *Koncepcja systemu* opisuje genezę powstania systemu OSF, jego główne cele i zadania oraz kluczowych interesariuszy, by przybliżyć nieco kontekst rozważań zawartych w dalszych rozdziałach;
2. Rozdział 2. *Procesy biznesowe* przedstawia opis najważniejszych procesów biznesowych (oraz stopień ich skomplikowania), jakie są wspierane przez system OSF, by Czytelnik zdawał sobie sprawę z jak bardzo rozbudowanym systemem mierzą się jego twórcy;
3. Rozdział 3. *Proces technologiczny* zaznajamia Czytelnika z cyklem technologicznym związanym z tworzeniem i rozwojem systemu OSF, w jego historycznym oraz obecnym kształcie;
4. Rozdział 4. *Architektura* jest najważniejszym rozdziałem tej publikacji; zawiera opis rozwiązań architektonicznych stosowanych w systemie historycznie i obecnie oraz na jakiej podstawie zostały podjęte decyzje o takim właśnie kształcie systemu.

ROZDZIAŁ 1

KONCEPCJA SYSTEMU

Urszula Sułek

Zintegrowany System Usług dla Nauki/Obsługa Strumieni Finansowania (skr. OSF) to system, którego głównym celem jest rejestrowanie i obsługa wniosków o finansowanie nauki wpływających do Ministerstwa Edukacji i Nauki (MEiN), Narodowego Centrum Nauki (NCN) i Narodowego Centrum Badań i Rozwoju (NCBR).

1.1. Tło historyczne

Organizacja i finansowanie nauki w Polsce pierwotnie zaczęły się kształtować jeszcze w okresie PRL. System ten jednak był niedofinansowany i nieefektywny. Polscy naukowcy często rozpoczynali badania nad pionierskimi rozwiązaniami, lecz z czasem brakowało im środków na rozbudowywanie aparatury i zwiększanie możliwości pomiarowych, wynikiem czego badania te (oraz ich sukcesy) były przejmowane przez zagraniczne laboratoria. Finansowanie badań podstawowych i stosowanych było nieefektywne. Podział dotacji budżetowych był w dużej mierze niesprawiedliwy na rzecz projektów z góry skazanych na porażkę oraz wdrożeń kosztownej aparatury przemysłowej. Polscy naukowcy nie mieli więc szans odnosić większych sukcesów na arenie międzynarodowej, reprezentując oczywiście rodzime instytucje badawcze [15]. System ten spotykał się z krytyką ze strony środowiska naukowego oraz ekspertów politycznych, najczęściej obcokrajowców oraz osób mających doświadczenie w roli ekspertów za granicą ([16]). Doprowadziło to ostatecznie do reformy nauki w Polsce, uwieńczonej ustawami o Komitecie Badań Naukowych – KBN [36], o szkolnictwie wyższym [35] oraz nowelizacją ustawy o jednostkach badawczo-rozwojowych [37].

Wskutek tej reformy, od 1991 roku finansowaniem badań naukowych w Polsce zajmowała się specjalnie powołana do tego celu instytucja – Komitet Badań Naukowych. Ustawa o KBN porządkowała cele, na jakie mogą być wydatkowane fundusze publiczne przeznaczone na finansowanie nauki, kryteria i tryb przyznawania i rozliczania środków finansowych ustalanych w budżecie państwa na naukę (w tym także sposób oceny wniosków o finansowanie) oraz sposób wykonywania nadzoru i kontroli realizacji badań naukowych i prac rozwojowych oraz innych zadań związanych z nauką [36].

W październiku 2004 roku na mocy Ustawy o zasadach finansowania nauki [38] Komitet Badań Naukowych został zlikwidowany, a jego zadania przekazano Radzie Nauki,

która powstała z przekształcenia KBN w ciało opiniodawczo-doradcze ministra właściwego do spraw nauki (wówczas ministra nauki i informatyzacji).

Wspomniana ustawa z 2004 roku weszła w życie, jednocześnie resort nauki podjął kroki zmierzające do obsługi wniosków przez system informatyczny. Zadaniem planowanego systemu miała być możliwość złożenia wniosku za pośrednictwem internetu oraz umożliwienie jego obsługi i przetwarzania po stronie ministerstwa.

Tak narodził się system OSF. W pierwszej jego wersji zaimplementowano wyłącznie moduł obsługi wniosków badawczych. Uruchomiono go jesienią 2006 roku. Pierwszym naborem był konkurs 33, termin składania wniosków określono na dzień 31 stycznia 2007 roku. Odpowiednik kamienia węgielnego, czyli inicjalny zapis (ang. *commit*) w repozytorium kodu źródłowego pojawił się 29 czerwca 2005 roku[1].

1.2. Główne cele i zadania systemu

Celem budowy i rozwoju systemu OSF jest wyeliminowanie w możliwie największym stopniu papierowego obiegu dokumentów w procesie obsługi wniosków o stypendia i granty naukowe przez agencje wykonawcze podległe ministerstwu właściwemu do spraw nauki, co w konsekwencji powoduje oszczędność czasu, nakładów i środków, a także zmniejszenie biurokracji związanej z tradycyjnym obiegiem dokumentów.

Główne zadania, jakie system realizuje, to:

- Umożliwienie przygotowywania i składania wniosków drogą elektroniczną;
- Ułatwienie dostępu do wersji roboczych wniosków dla współredaktorów;
- Dokonywanie oceny formalnej oraz merytorycznej wniosków w sposób niewymagający przesyłania dokumentacji papierowej pomiędzy uczestnikami procesu, w tym do recenzentów zagranicznych;
- Skrócenie czasu przygotowywania, negocjacji i podpisywania umów dwu- oraz wielostronnych, a także aneksów i porozumień;
- Umożliwienie składania drogą elektroniczną raportów dotyczących realizacji projektów badawczych, wsparcie procesu kontrolowania i rozliczania przyznanego finansowania.

[1] Informacje dotyczące historii powstawania systemu pochodzą z dokumentacji projektowej oraz kodu źródłowego systemu OSF

1.3. Interesariusze

Jak już wspomniano we wcześniejszych akapitach, głównym interesariuszem projektu budowy i rozwoju systemu OSF jest ministerstwo właściwe do spraw nauki. W 2004/05 roku resort ten nosił nazwę Ministerstwa Nauki i Informatyzacji, obecnie (w roku 2021) – Ministerstwa Edukacji i Nauki.

Kolejnym interesariuszem wykorzystującym system OSF do celów finansowania nauki w zakresie realizacji badań stosowanych jest powołane latem 2007 roku Narodowe Centrum Badań i Rozwoju. NCBR jest agencją wykonawczą, realizującą zadania z zakresu polityki naukowej, naukowo-technicznej i innowacyjnej państwa, umożliwiającą współpracę środowiska naukowego z biznesem.

Ostatnią (na dzień dzisiejszy, tj. w ostatnim kwartale 2021 roku) z instytucji, która wykorzystuje system do analogicznych celów jest Narodowe Centrum Nauki, powołane do życia na mocy ustawy z dnia 30 kwietnia 2010 roku o Narodowym Centrum Nauki. NCN jest agencją wykonawczą, finansującą projekty badawcze z zakresu badań podstawowych, stypendia doktorskie oraz staże po uzyskaniu stopnia naukowego doktora.

Wymienione wyżej agencje były sukcesywnie włączane do obsługi wniosków w systemie informatycznym OSF. Z uwagi na fakt, że każda z agencji wykonawczych oraz ministerstwo tworzą odrębne instytucje, o odrębnych celach, zasadach funkcjonowania i budżetach, z biznesowego punktu widzenia w systemie informatycznym wydzielone zostały osobne moduły (chociaż biorąc pod uwagę architekturę, jak się później dowiemy, moduły te nie były z początku wyraźnie wyodrębnione). Każda z instytucji, a konkretnie ich pracownicy, ma dostęp tylko do swoich wniosków i obsługuje je na własnych zasadach, definiując wymagania do spełnienia w systemie informatycznym.

W tym miejscu należałoby wspomnieć o beneficjentach systemu finansowania nauki. Są to przede wszystkim naukowcy, indywidualni bądź wchodzący w skład zespołów projektowych powoływanych przez podmioty będące głównie instytucjami szkolnictwa wyższego oraz jednostkami naukowo-badawczymi lub grupami podmiotów prowadzącymi działalność naukową, otrzymujący stypendia bądź granty naukowe na skutek pomyślnego przejścia procesu oceny składanych za pośrednictwem systemu OSF wniosków.

Wreszcie, z systemu korzystać mogą również eksperci – recenzenci wniosków, krajowi bądź zagraniczni.

ROZDZIAŁ 2

PROCESY BIZNESOWE

Urszula Sułek

Główne procesy biznesowe (por. Ilustracja 2.1), realizowane w systemie OSF, przedstawiają się następująco[2]:

1. Wnioskowanie;
2. Ocena wniosków;
3. Obsługa finansowania;
4. Rozliczanie finansowania.

Ilustracja 2.1. Główne procesy biznesowe realizowane w systemie OSF

Charakterystyka wymienionych wyżej procesów biznesowych zamieszczona jest w podrozdziałach poniżej.

OSF jest w chwili obecnej bardzo rozbudowanym pod kątem biznesowym systemem. Przez kilkanaście lat od jego powstania twórcy systemu mierzyli się z problemem dołączania kolejnych instytucji obsługujących różne rodzaje grantów. W pierwszej kolejności obsługiwane były wnioski Ministerstwa Nauki i Szkolnictwa Wyższego (obecnie Ministerstwo Edukacji i Nauki), a w następnej kolejności dołączane były wnioski Narodowego Centrum Badań i Rozwoju, w końcu Narodowego Centrum Nauki. Każda z tych instytucji bazuje na odrębnych regulaminach przyznawania i rozliczania stypendiów i grantów naukowych. Ponieważ instytucje te uruchamiają różne konkursy, o różnych kryteriach i trybach przyznawania finansowania oraz z różną częstotliwością, system OSF musi być na bieżąco dostosowywany do warunków kolejnych uruchamianych w nim konkursów. Z tego względu, implementacja procesów biznesowych w systemie musi być na tyle elastyczna, by zespół wytwórczy mógł w trybie ciągłym dostosowywać się do zmiennych warunków dyktowanych przez instytucje finansujące działania naukowe. Stąd też,na

[2] Opracowanie własne na podstawie dokumentacji projektowej systemu OSF

chwilę obecną, istnieje wiele wariantów przebiegu wymienionych tutaj procesów biznesowych, zależnych od instytucji oraz warunków obsługiwanego konkursu.

O skali trudności w budowie i utrzymaniu OSF, jaką stanowi rozbudowany wachlarz możliwych konfiguracji procesów biznesowych, świadczy następujące zestawienie liczby obsługiwanych konkursów przez poszczególne instytucje od początku funkcjonowania systemu[3]:

- MEiN – 106;
- NCBR – 68;
- NCN – 164.

2.1. Wnioskowanie

Wnioski w konkursach obsługiwanych przez MEiN, NCN i NCBR składane są wyłącznie w formie elektronicznej. Proces wnioskowania w systemie OSF przebiega podobnie dla wszystkich wymienionych wyżej instytucji, z niewielkimi różnicami (por. Ilustracja 2.2).

Ilustracja 2.2. Etapy procesu wnioskowania

Proces wnioskowania rozpoczyna się od uruchomienia naboru wniosków w konkursie przez daną instytucję. Ze względów formalnych uruchomienie naboru musi być poprzedzone ogłoszeniem o jego otwarciu z planowaną datą rozpoczęcia naboru wniosków oraz warunkami danego konkursu (edycji). Ogłoszenie takie zazwyczaj publikowane jest na stronach internetowych instytucji. Zaś możliwość utworzenia wniosku w danym naborze kontrolowana jest już przez system informatyczny OSF – jest to możliwe tylko w określonym przez regulamin okresie czasu. Dla wybranych konkursów nabór jest prowadzony w trybie ciągłym.

Utworzenie wniosku w systemie OSF jest możliwe tylko dla zarejestrowanych użytkowników. Redaktor wniosku może zaprosić do współredakcji wniosku również redaktorów pomocniczych, a także dodać czytelnika wniosku (użytkownika, który ma prawo tylko do czytania treści wniosku, bez możliwości wprowadzania jakichkolwiek zmian). Wszystkie te osoby muszą mieć zarejestrowane w systemie konta użytkownika. Nie ma możliwości, by anonimowe osoby mogły przeglądać i edytować treść przygotowywanej w systemie OSF dokumentacji. Istotne jest również, że każda operacja wykonywana na wnioskach (a także innych dokumentach obsługiwanych przez system)

[3] Stan na dzień 01.10.2021 r.

zapisywana jest w systemie z datą i godziną wystąpienia oraz danymi użytkownika, który tę operację wykonał.

Każdy wniosek utworzony w systemie OSF opatrzony jest metryką, tak zwanym nagłówkiem wniosku. Najważniejszymi informacjami opisującymi wniosek są: identyfikator, numer rejestracyjny wniosku wraz z datą wpłynięcia (dostępne po zarejestrowaniu wpłynięcia wniosku do instytucji), dane opisujące edycję konkursu, status, dane opiekuna po stronie instytucji, dane opisujące tytuł projektu, kierownika oraz podmiot realizujący projekt.

Niewątpliwą zaletą wypełniania wniosków o granty w systemie OSF jest fakt, iż użytkownik może zapisać edytowany wniosek i powrócić do jego edycji w dowolnym momencie, zanim zdecyduje się go finalnie wysłać do instytucji. Jest to szczególnie ważne ze względu na fakt, iż wnioski o finansowanie nauki są w większości bardzo rozbudowane. Formularze zawierają średnio od kilku do kilkunastu sekcji do wypełnienia wymaganymi w danym konkursie danymi oraz załącznikami w postaci plików. Wypełnienie takiego rozbudowanego wniosku jednorazowo jest często niemożliwe.

Drugą ważną zaletą wnioskowania w formie elektronicznej jest automatyczne walidowanie pól formularza na bieżąco, w trakcie wypełniania wniosku. Redaktor wniosku uzyskuje podpowiedzi odnośnie prawidłowego wprowadzania danych w formularzu oraz informacje o brakach koniecznych do uzupełnienia. Skraca to znacząco sam proces przygotowywania dokumentu oraz pozwala zapobiec wielu błędom formalnym we wczesnym etapie, jeszcze zanim wniosek trafi do oceny do odpowiedniej instytucji. Powoduje to jednak dodatkowe skomplikowanie implementacji wniosku w systemie. Na formularze nałożona jest ogromna liczba reguł biznesowych i walidacyjnych, specyficznych dla każdej edycji konkursu danego rodzaju wniosku.

WNIOSKI NCN

W systemie OSF obsługiwane są aktualnie cztery rodzaje wniosków, których odbiorcą jest Narodowe Centrum Nauki:

- Wnioski o finansowanie projektu badawczego z zakresu badań podstawowych;
- Wnioski o przyznanie środków finansowych na realizację działania naukowego;
- Wnioski w konkursach międzynarodowych w ramach inicjatyw dwu- i wielostronnych, których ocena merytoryczna odbywa się poza NCN;
- Wnioski o finansowanie komponentów badawczych w projektach finansowanych w ramach programów organizowanych przez Narodową Agencję Wymiany Akademickiej.

W każdym z wymienionych wyżej typów wniosków występuje bardziej szczegółowy podział[4].

Wnioski o finansowanie projektu badawczego z zakresu badań podstawowych mają największą liczbę wariantów zależnych od tego, kto realizuje projekt badawczy lub jaki jest cel tego projektu:

- OPUS – otwarte dla wszystkich naukowców;
- PRELUDIUM – realizowane przez osoby nieposiadające stopnia naukowego doktora;
- PRELUDIUM BIS – realizowane przez doktorantów w szkołach doktorskich;
- SONATINA – realizowane przez osoby posiadające stopień naukowy doktora, uzyskany w okresie do 3 lat przed rokiem wystąpienia z wnioskiem;
- SONATA – realizowane przez osoby posiadające stopień naukowy doktora;
- SONATA BIS – mające na celu powołanie nowego zespołu naukowego, realizowane przez osoby posiadające stopień naukowy lub tytuł naukowy, które uzyskały stopień naukowy doktora w okresie od 5 do 12 lat przed rokiem wystąpienia z wnioskiem;
- MAESTRO – dla doświadczonych naukowców na projekty badawcze mające na celu realizację pionierskich badań naukowych, w tym interdyscyplinarnych, ważnych dla rozwoju nauki, wykraczających poza dotychczasowy stan wiedzy, których efektem mogą być odkrycia naukowe;
- ETIUDA – na stypendia doktorskie;
- UWERTURA – na staże w zagranicznych zespołach naukowych realizujących granty ERC[5];
- GRIEG – na projekty badawcze w ramach Norweskiego Mechanizmu Finansowego 2014-2021;
- IDEALAB – na przełomowe, interdyscyplinarne projekty badawcze finansowane w ramach Mechanizmu Finansowego Europejskiego Obszaru Gospodarczego 2014-2021;
- POLS – realizowane przez naukowca przyjeżdżającego z zagranicy w ramach Norweskiego Mechanizmu Finansowego 2014-2021;
- BEETHOVEN – na polsko-niemieckie projekty badawcze;
- DAINA – na polsko-litewskie projekty badawcze;
- SHENG – na polsko-chińskie projekty badawcze;
- OPUS LAP – na projekty badawcze realizowane w ramach inicjatywy WEAVE[6];
- POLONEZ BIS – na projekty badawcze realizowane przez naukowców przyjeżdżających z zagranicy.

[4] Opis obsługiwanych przez system OSF konkursów NCN przygotowany został na podstawie spisu konkursów zamieszczonego na stronie internetowej Narodowego Centrum Nauki oraz dokumentacji projektowej systemu OSF, stan na koniec III kwartału 2021

[5] European Research Council, ERC – Europejska Rada ds. Badań Naukowych

[6] WEAVE – program wielostronnej współpracy ukierunkowany na finansowanie międzynarodowych projektów badawczych o wyróżniającym się poziomie naukowym, finansowany przez 12 europejskich instytucji finansujących badania naukowe. Strona internetowa inicjatywy: weave-research.net (dostęp: 17.08.2021)

Wnioski o przyznanie środków finansowych na realizację działania naukowego MINIATURA nie mają podtypów, ale podobnie jak inne rodzaje wniosków, powtarzające się cyklicznie edycje konkursu.

Wśród wniosków w konkursach międzynarodowych w ramach inicjatyw dwu- i wielostronnych, których ocena merytoryczna odbywa się poza NCN można zaś wymienić wiele programów realizowanych we współpracy NCN z innymi instytucjami finansującymi projekty badawcze, głównie europejskimi. Przede wszystkim wyróżnić należy w tej grupie konkurs MOZART na polsko-austriackie projekty badawcze, przy współpracy z agencją FWF[7]. Konkurs ALPHORN na polsko-szwajcarskie projekty badawcze realizowany jest wspólnie z SNSF[8]. Przykładem wielostronnej współpracy różnych agencji europejskich jest program CEUS. Realizowany jest on w ramach współpracy między NCN i agencjami z Austrii (wspomniana wcześniej agencja FWF), Czech (GAČR – Czech Science Foundation – Grantová agentura České Republiky) oraz Słowenii (ARRS – Slovenian Research Agency – Javna agencija za raziskovalno dejavnost Republike Slovenije). Finansowanie projektów z zakresu badań podstawowych we wszystkich dyscyplinach naukowych, planowanych do realizacji we współpracy zespołów badawczych z dwóch lub trzech krajów uczestniczących w programie możliwe było poprzez uruchomienie konkursu CEUS-UNISONO. NCN związany jest także z międzynarodowymi konsorcjami oraz sieciami skupiającymi różne agencje finansujące badania naukowe, powołanymi w celu wspierania badań naukowych z różnorakich dziedzin. Przykładem takiego konsorcjum jest CHIST-ERA[9], które w ramach sieci ERA-NET Cofound wspiera badania z zakresu technologii informacyjnych oraz komunikacyjnych ICST (Information and Communication Science and Technologies) w ramach konkursów CHIST-ERA. Inną siecią, do której należy NCN jest JPI Cultural Heritage. Wspiera ona badania naukowe na temat dziedzictwa kulturowego w ramach konkursu JPI. Współpraca międzynarodowa Narodowego Centrum Naukowego stale się rozwija i systematycznie pojawiają się kolejne rodzaje konkursów międzynarodowych, także obsługiwane przez system OSF.

Formularze wniosków przekazywane do NCN składają się z zestawu sekcji, wymaganych do uzupełnienia w różnej konfiguracji i w różnym stopniu – zależnym od rodzaju wniosku i wymogów danej edycji konkursu. Sekcja wniosku zawiera zestaw powiązanych ze sobą tematycznie pól do uzupełnienia przez wnioskodawcę. Zespół wytwórczy, we współpracy z przedstawicielami Narodowego Centrum Nauki, stara się ujednolicać wygląd wniosków finansowanych przez tę instytucję. Dla użytkowników końcowych, to jest redaktorów, czytelników, a w dalszych etapach przetwarzania również pracowników NCN, recenzentów i innych ekspertów mających wgląd w formularze, jest dużym ułatwieniem kiedy mogą skupić się na merytoryce zawartej we wnioskach zamiast

[7] FWF jest skrótowym określeniem austriackiej agencji Austrian Science Fund, od oryginalnej nazwy: Fonds zur Förderung der wissenschaftlichen Forschung. Strona internetowa instytucji: fwf.ac.at/de (dostęp: 17.08.2021)
[8] Swiss National Science Foundation, SNSF – szwajcarska instytucja finansująca projekty badawcze we wszystkich dyscyplinach naukowych . Strona internetowa instytucji: snf.ch/en (dostęp: 17.08.2021)
[9] CHIST-ERA – European Coordinated Research on Long-term Challenges in Information and Communication Sciences & Technologies. Strona internetowa konsorcjum: chistera.eu (dostęp: 17.08.2021)

na układzie formularza, który jest spójny z wcześniejszymi konkursami i nie stanowi elementu zaskoczenia dla czytającego, który poszukuje konkretnych informacji.

Wypełnianie wniosku w ramach modułu NCN kończy się w momencie zablokowania formularza, po wcześniejszym upewnieniu się, że walidacja całego dokumentu przebiegła poprawnie. W zależności od rodzaju wypełnianego wniosku, przed wysłaniem do instytucji wymagane mogą być jeszcze dodatkowe kroki, takie jak dołączenie wymaganych załączników (na przykład spersonalizowane potwierdzenie złożenia wniosku, pobierane z systemu OSF, które muszą zostać podpisane odręcznie lub elektronicznie i załączone zwrotnie do wniosku, czy też upoważnienia do reprezentowania podmiotu).

Zablokowany wniosek można wycofać do edycji, jeśli redaktor uzna, że jego treść wymaga zmodyfikowania przed jego złożeniem.

Kompletna, ostateczna wersja wniosku wraz z wymaganymi załącznikami może zostać wysłana do Narodowego Centrum Nauki. Wysłanie wniosku oznacza, że dokument został oficjalnie złożony i od tej chwili rozpoczyna się jego procesowanie po stronie instytucji. Instytucja ta honoruje wnioski przesyłane jedynie w wersji elektronicznej, za pośrednictwem systemu OSF.

Po wysłaniu wniosku, jest on automatycznie rejestrowany w systemach wewnętrznych NCN, gdzie nadawany jest numer rejestracyjny dla dokumentu.

WNIOSKI NCBR

Konkursy Narodowego Centrum Badań i Rozwoju przeprowadzane w systemie OSF w ostatnich kilku latach można podzielić na dwie zasadnicze grupy. Pierwszą stanowią konkursy związane z obronnością i bezpieczeństwem państwa (ozn. DOB), drugą natomiast – konkursy cywilne (nieobronne).

W ramach pierwszej grupy możemy wymienić między innymi kolejne edycje konkursów z programu SZAFIR, przeznaczone dla uczelni, instytutów naukowych PAN, instytutów badawczych, konsorcjów naukowych, centrów naukowo-przemysłowych oraz przedsiębiorców i przeznaczeniem których jest stworzenie odpowiednich warunków organizacyjno-finansowych w celu pobudzenia inicjatywy i wykorzystania potencjału instytucji naukowych i przedsiębiorców służących wykreowaniu nowych innowacyjnych pomysłów rozwijania technologii i techniki w obszarze bezpieczeństwa i obronności państwa.

Oprócz zawartości formularzy i tematyki konkursów, wnioski tych grup różnią się od siebie przede wszystkim w zakresie obsługi warunków finansowych. Charakterystyczne dla konkursów nieobronnych są znacznie bardziej rozbudowane warunki kontroli w części finansowej wniosków, związane m.in. z możliwością wnioskowania w nich o pomoc publiczną i *de minimis*.

Dość charakterystyczne dla obsługi wniosków NCBR jest to, że złożone przez wnioskodawców dokumenty od razu są uznawane za przyjęte. Nie można ich wycofać do wnioskodawców. Pracownicy NCBR nie widzą formularzy w trakcie ich przygotowania przez wnioskodawców. W module wniosków obronnych (DOB) procesowanie dokumentów przez instytucję zaczyna się od przypisania opiekunów do złożonych wniosków. Podobny mechanizm jednak nie występuje w pozostałych modułach wykorzystywanych przez NCBR.

WNIOSKI MEIN

Ministerstwo Edukacji i Nauki (a wcześniej Ministerstwo Nauki i Szkolnictwa Wyższego) przyznaje środki finansowe na cele związane z nauką za pomocą różnego rodzaju programów. W wielu przypadkach ogłaszane są konkursy pojedyncze, niepowtarzalne. Duża grupa konkursów ma nabór otwarty przez cały rok. Istnieją też takie, które uruchamiane są cyklicznie w kolejnych edycjach. Można tutaj wyróżnić najbardziej popularne wnioski o przyznanie dotacji na inwestycje związane z działalnością naukową, na rozbudowę infrastruktury badawczej lub informatycznej, ale także związane ze stypendiami dla studentów i młodych naukowców oraz rozwojem sportu akademickiego.

Formularz wniosku w ramach MEiN, podobnie jak w innych modułach, składa się z sekcji. Część z nich jest wspólna dla wszystkich rodzajów wniosków, a pozostałe uzależnione od specyfiki danego konkursu.

Redaktorzy wniosków mają możliwość wyboru, w jaki sposób wniosek zostanie wysłany do resortu. Występują trzy rodzaje wysyłki wniosku, tj. elektronicznie: poprzez ePUAP[10], z podpisem cyfrowym lub papierowo (pocztą tradycyjną). Niezależnie od wyboru opcji wysyłki, wniosek należy również wysłać w systemie OSF. Dostępne opcje są zależne od wymagań ministerstwa w danym konkursie, tj. mogą występować do wyboru wszystkie trzy wymienione wyżej opcje albo ich podzbiór.

Dla każdego wniosku możliwe jest wygenerowanie zawartości całego formularza w postaci pliku PDF, zarówno wydruku oficjalnego, jak i roboczego.

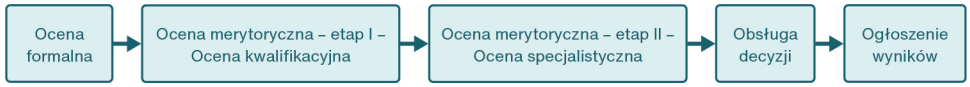
[10] ePUAP – Elektroniczna Platforma Usług Administracji Publicznej.

2.2. Ocena wniosków

Przebieg procesu oceny wniosków zależny jest od instytucji. Choć wspólnym dla każdej z trzech obsługiwanych instytucji jest podział tego procesu na ocenę formalną i merytoryczną, każda z nich rządzi się swoimi prawami i uwarunkowaniami[11].

OCENA WNIOSKÓW W NCN

System OSF wspiera proces oceny wniosków w Narodowym Centrum Nauki poprzez umożliwienie wprowadzania informacji o kolejnych etapach oceny poszczególnych wniosków.



Ilustracja 2.3. Etapy procesu oceny wniosków w Narodowym Centrum Nauki

Ocena wniosków w Narodowym Centrum Nauki jest wieloetapowa i rozbudowana (por. Ilustracja 2.3). W NCN za procesowanie wniosku od momentu jego rejestracji do zakończenia obsługi odpowiada wyznaczony opiekun (jeden z pracowników instytucji przypisywany w systemie do wniosku). Opiekun wniosku dokonuje pierwszej oceny dokumentu pod kątem formalnym. Wprowadza on do systemu informacje o wyniku swojej oceny, to jest czy wniosek jest wypełniony poprawnie lub czy też zawiera jakieś uchybienia. Tę informację przekazuje do oceny formalnej koordynatora (osoba wyznaczona do koordynacji zespołu ekspertów z danej dziedziny). Koordynator, uwzględniając uwagi opiekuna wniosku, podejmuje decyzję odnośnie odrzucenia wniosku lub jego odesłania do poprawy, jeśli ten zawiera braki, które możliwe są jeszcze do uzupełnienia na tym etapie. Jeśli zaś ocena koordynatora jest pozytywna, wniosek może być dalej procedowany. Wówczas, zależnie od zasad oceny wniosków danego typu, określonych w regulaminie przyznawania środków w danym zakresie, wniosek może od razu zostać przekierowany do etapu finansowania (tak się dzieje na przykład we wnioskach współfinansowanych z Narodową Agencją Wymiany Akademickiej). Większość naborów wymaga jednak przeprowadzenia oceny merytorycznej wniosków przed wyłonieniem zwycięskiej listy projektów zakwalifikowanych do finansowania.

Ocena merytoryczna wniosku składa się z dwóch etapów. Pierwszy to ocena kwalifikacyjna wniosków w danym konkursie, drugi – ocena specjalistyczna. Ocena merytoryczna wykonywana jest przez ekspertów dziedzinowych. W pierwszym etapie są to eksperci wyznaczeni z tzw. zespołów ekspertów, w drugim zaś recenzenci zewnętrzni. Każdy etap oceny kończy się ustaleniami dokonywanymi przez członków zespołu

[11] Opis procesu oceny wniosków został opracowany na podstawie dokumentacji projektowej systemu OSF.

ekspertów podczas posiedzeń panelowych. W wyniku prac ekspertów powstają ostateczne listy rankingowe projektów zakwalifikowanych do finansowania.

W trakcie etapu oceny kwalifikacyjnej wykonywane są ekspertyzy (recenzje, czy też opinie) wniosku, wprowadzane do systemu przez ekspertów dziedzinowych. Każdy wniosek jest oceniany przez dwóch niezależnych ekspertów. By taka ocena była możliwa w systemie OSF, wspiera on takie czynności jak wyznaczanie zespołu ekspertów (czyli grupy ekspertów dziedzinowych właściwej dla danego konkursu), przypisywanie ich do tzw. paneli składających się z przedstawicieli jednej z dziedzin naukowych oraz superpaneli składających się z przedstawicieli różnych dziedzin, wreszcie zapraszanie konkretnych ekspertów do wykonania ekspertyzy. Wyznaczony ekspert otrzymuje dostęp do przydzielonych wniosków i formularza oceny. W kolejnym kroku ekspert wykonuje ekspertyzę wniosku i wprowadza jej wynik do systemu. Poszczególne zdarzenia są zapisywane w systemie, co umożliwia pracownikom NCN określenie poziomu zaawansowania wykonania ekspertyzy na każdym etapie. Po wprowadzeniu opinii wniosku przez wszystkich zaproszonych ekspertów, system OSF automatycznie wylicza ocenę końcową opinii. Dokonuje tego na podstawie odpowiedzi wprowadzonych przez eksperta w formularzu opinii dla danego wniosku. Istotne jest także, że dany ekspert może również na tym etapie zarekomendować wykonanie ekspertyzy przez zewnętrznego recenzenta (która będzie możliwa w kolejnym etapie oceny merytorycznej). W trakcie pierwszego posiedzenia panelu ekspertów omawiają i uzgadniają oni oceny końcowe wniosków, odrzucając lub kwalifikując wnioski do II etapu oceny merytorycznej poprzez uzupełnienie formularza oceny panelu. Dla wniosków niezakwalifikowanych do II etapu wydawane są decyzje odmowne. Wszystkie powyższe czynności wspierane są przez system OSF.

Drugi etap oceny merytorycznej, czyli ocena specjalistyczna, polega na dokonaniu oceny wniosków przez wyznaczonych recenzentów zewnętrznych. Zewnętrzni eksperci, podobnie jak w pierwszym etapie, wypełniają w systemie formularze oceny wniosku. W niektórych programach wymagana jest dodatkowo rozmowa eksperta z kierownikiem projektu. Na drugim posiedzeniu zespołu ekspertów, po otrzymaniu recenzji ponownie dokonywane jest uzgodnienie ocen wniosków przez zespół ekspertów, w wyniku czego powstaje lista rankingowa wniosków zakwalifikowanych do finansowania. Wnioski niezakwalifikowane do finansowania otrzymują decyzje odmowne.

System OSF umożliwia członkom zespołu ekspertów dostęp do wniosków, formularzy recenzji, wykonanych recenzji oraz do listy rankingowej. Możliwe jest także określanie granicy zakwalifikowanych do finansowania wniosków. Oceny wniosków uzgodnione w trakcie spotkań panelowych wprowadzane są do systemu wraz z uzasadnieniem, co sprawia, że wszystkie czynności prowadzące do ostatecznej decyzji są transparentne.

Decyzje dotyczące finansowania wniosków obsługiwane są w systemie w formie rejestracji daty podpisania decyzji oraz daty odbioru decyzji – niezależnie od tego, czy decyzja była pozytywna czy negatywna. W przypadku decyzji negatywnej, każdy wnioskodawca ma prawo odwołania się od decyzji. System wspiera przeprowadzenie procedury odwoławczej, umożliwiając zarejestrowanie daty odwołania od decyzji. Takie wnio-

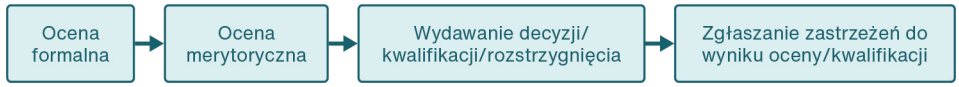
ski mogą być skierowane z powrotem na posiedzenie komisji, może zostać wykonana dodatkowa recenzja i wprowadzana jest decyzja odnośnie otrzymanego odwołania.

Ogłoszenie wyników odbywa się już poza systemem. Lista wniosków, które decyzją NCN otrzymują finansowanie wraz z kwotą finansowania, publikowana jest na stronie internetowej NCN.

OCENA WNIOSKÓW W MEIN

Podstawowy model oceny wniosków MEiN zdefiniowany w systemie OSF można podzielić na następujące etapy biznesowe (por. Ilustracja 2.4):

1. Ocena formalna;
2. Ocena merytoryczna (w zależności od konkursu jedno- lub wieloetapowa, z możliwością składania zastrzeżeń lub bez);
3. Wydawanie decyzji/kwalifikacji/rozstrzygnięcia (nazwa może się różnić w zależności od programu/konkursu) oraz udostępnianie wyniku oceny i decyzji bądź kwalifikacji;
4. Możliwość złożenia zastrzeżenia do wyniku oceny bądź kwalifikacji.



Ilustracja 2.4. Etapy procesu oceny wniosków w Ministerstwie Edukacji i Nauki

Wnioski wysłane do MEiN są po stronie ministerstwa odbierane w systemie OSF. Podczas odbioru przydzielany jest opiekun wniosku, nadawany jest numer rejestracyjny wniosku oraz podawany jest numer RPW[12], a także numer sprawy w EZD[13]. Wniosek można również wycofać ponownie do edycji redaktora.

Pierwszym etapem oceny wniosku, który został wysłany do ministerstwa jest ocena formalna wniosku. Pracownik urzędu weryfikuje, czy wniosek jest kompletny. W standardowych wnioskach (to jest takich, dla których zostały wypracowane wspólne rozwiązania, niezależnie od programu/konkursu) zwykle pracownik oznacza kompletność wniosku wybierając tylko odpowiedni przycisk, natomiast w starszych wnioskach np. wnioskach na realizację projektu międzynarodowego, należy wypełnić formularz oceny kompletności. Jeśli wniosek nie jest kompletny, to zostaje odesłany do uzupełnienia do redaktora wniosku. Podczas odesłania należy wskazać sekcje dokumentu, które

[12] RPW – Rejestr Przesyłek Wpływających, rejestr służący do ewidencjonowania w kolejności chronologicznej przesyłek otrzymywanych przez podmiot [29]
[13] EZD – Elektroniczny System Zarządzania Dokumentacją, system teleinformatyczny umożliwiający wykonywanie w nim czynności kancelaryjnych, dokumentowanie przebiegu załatwiania spraw oraz gromadzenie i tworzenie dokumentów elektronicznych [29]

będą wymagały uzupełnienia. Do czasu rozpoczęcia edycji wniosku przez redaktora, pracownik może wycofać wniosek z uzupełnień. Uzupełnione Wnioski redaktor wysyła ponownie do ministerstwa. Termin na wysłanie poprawionego wniosku może się różnić pomiędzy różnymi konkursami. W systemie istnieje również opcja rezygnacji wnioskodawcy, dostępna na każdym etapie obsługi do momentu podpisania umowy. Proces ten jest standardowy, dostępny we wszystkich wnioskach MEiN.

Kolejnym etapem obsługi wniosku wysłanego do Ministerstwa Edukacji i Nauki jest ocena merytoryczna wniosku. System obsługuje trzy scenariusze oceny:

- Ocena merytoryczna pełna;
- Ocena merytoryczna uproszczona;
- Ocena merytoryczna prosta.

Ocena merytoryczna pełna zawiera najbardziej rozbudowaną ścieżkę oceny merytorycznej dla wniosków ministerialnych. Składa się z oceny referentów, oceny eksperta, oceny końcowej oraz rozstrzygnięcia. Dla każdej z wyżej wymienionych ocen dostępne są inne kryteria oceny oraz dla każdego z kryteriów trzeba wskazać liczbę punktów.

Ocena merytoryczna uproszczona różni się od oceny pełnej przydzieleniem eksperta zamiast referentów oraz brakiem oceny zbiorczej referentów w początkowej fazie procesu. Ta forma oceny składa się z oceny eksperta, oceny końcowej oraz rozstrzygnięcia. Podobnie jak w przypadku oceny pełnej, dla każdej z wymienionych wyżej ocen dostępne są inne kryteria oceny, a także dla każdego z kryteriów trzeba wskazać liczbę punktów. W ramach oceny merytorycznej uproszczonej oraz pełnej funkcjonuje udostępnianie oceny dla redaktora wniosku.

Ocena merytoryczna prosta różni się od oceny pełnej i uproszczonej brakiem oceny końcowej (najczęściej zbiorczej oceny referentów) oraz udostępnieniem oceny i zgłaszaniem zastrzeżeń przez redaktora wniosku. Składa się z oceny referenta oraz rozstrzygnięcia. W systemie podawana jest pozycja na liście rankingowej, natomiast sama lista tworzona jest poza systemem OSF.

Oprócz opisanych wyżej trzech standardowych ścieżek występują także inne, niestandardowe. Podobnie do obsługi wniosków w NCN, w ścieżkach niestandardowych na etapie oceny merytorycznej do wniosków dodawani są eksperci dokonujący recenzji wniosków. W zależności od rodzaju wniosku liczba ekspertów może się różnić. Dodani we wniosku eksperci uzupełniają formularz oceny, wskazując liczbę punktów w ramach każdego dodanego we wniosku osiągnięcia. Również, dla wybranych konkursów w systemie dostępna jest lista rankingowa. Proces obsługi oceny merytorycznej w tej formie kończy decyzja oraz udostępnienie decyzji wnioskodawcom, a także opcjonalnie, w zależności od konkursu, możliwość składania zastrzeżeń do wyniku oceny bądź kwalifikacji.

Jeden z rodzajów wniosków realizowanych w ramach MEiN funkcjonuje w sposób charakterystyczny. Wniosek składany jest dwuetapowo, to znaczy po zakwalifikowaniu do pierwszego etapu oceny redaktorzy muszą wypełnić drugą, niedostępną wcześniej

część wniosku. Przykład ten pokazuje, jak bardzo różne mogą być przebiegi procesów o podobnych celach realizowanych przez system OSF, który musi być przygotowany do tego, by móc obsługiwać różne tzw. odstępstwa od reguły.

OCENA WNIOSKÓW W NCBR

System OSF wspiera proces oceny wniosków NCBR w podziale na następujące etapy biznesowe (por. Ilustracja 2.5):

1. Ocena formalna (wraz z obsługą poprawy załączników wniosku);
2. Ocena merytoryczna (w zależności od konkursu jedno- lub wieloetapowa);
3. Wydawanie i generowanie decyzji oraz publikacja wyników.

Ilustracja 2.5. Etapy procesu oceny wniosków w Narodowym Centrum Badań i Rozwoju

Pomiędzy poszczególnymi konkursami ogłaszanymi przez Narodowe Centrum Badań i Rozwoju istnieją pewne różnice merytoryczne w sposobie obsługi poszczególnych etapów oceny wniosków. Często wynikają one ze specyfiki konkursu. Sposób oceny wniosków każdorazowo jest definiowany przez NCBR w ramach Regulaminu konkursu, biorąc pod uwagę charakter konkursu, dostępność ekspertów z danej dziedziny czy wymogi ministerstw, na poczet których będą realizowane projekty badawcze. Sposoby oceny niektórych etapów nawet w obrębie kolejnych edycji tego samego typu konkursu, mogą też ulegać modyfikacjom.

W obrębie oceny formalnej, NCBR wyróżnia się koniecznością dokładnej rejestracji wyników kontroli formalnej wniosku. Sam proces oceny formalnej stosowany w ramach wniosków NCBR z lat 2016-2021 jest jednak dość uniwersalny, gdyż podobne rozwiązanie jest stosowane praktycznie we wszystkich konkursach.

Każdorazowo pracownik NCBR będący opiekunem wniosku musi wypełnić dość rozbudowany formularz oceny formalnej, składający się z kilku kryteriów. Mimo iż treść kryteriów różni się w zależności od konkursu, weryfikowane są te same aspekty formalne wniosków. Standardowo w ramach oceny formalnej rejestrowane jest także tzw. podsumowanie oceny – pytania podsumowujące przeprowadzoną kontrolę wniosku. Decydują one o tym, czy wniosek kierowany jest do dalszej oceny, uznawany za niepoprawny formalnie, czy ma podlegać uzupełnieniu i poprawie. System rejestruje dane osoby, która uzupełnia formularz bądź umożliwia rejestrację danych osób, które jej dokonują, gdy konieczna jest ocena przez dwóch pracowników NCBR.

Model oceny formalnej stosowany w NCBR mocno ograniczał do tej pory bezpośredni udział wnioskodawcy (redaktora wniosku) w systemie w momencie skierowania wniosku do uzupełnienia. Poprawa wniosku obejmuje w NCBR jedynie możliwość podmiany za-

łączników i zmiany treści złożonych przez wnioskodawcę oświadczeń. W związku z tym, zmiany we wnioskach były dokonywane przez pracowników NCBR, na podstawie materiałów przekazanych im przez wnioskodawców poza systemem. Wnioskodawca mógł jedynie akceptować zmiany dokonane przez opiekuna wniosku. W ostatnim czasie można jednak zaobserwować zmianę podejścia NCBR w tym aspekcie – tzn. chęć odwrócenia ról, tak aby to redaktor wniosku sam modyfikował treść wniosku. Oznacza to zbliżenie się do modelu oceny stosowanego przy ocenie formalnej wniosków w MEiN. Dodatkowo system weryfikuje termin na skorygowanie wniosku i odbiera uprawnienia edycji wniosku po jego przekroczeniu.

Ocena formalna kończy się w NCBR z chwilą jej zatwierdzenia przez opiekuna wniosku o zaawansowanych uprawnieniach. Krok ten jest nieodwracalny, co stoi w kontraście do MEiN, gdzie można w systemie cofnąć podjętą czynność związaną z podaniem ostatecznego wyniku oceny formalnej.

Ocena merytoryczna wniosków NCBR jest mocno zróżnicowana w zależności od typu konkursu i jego edycji. Występują tutaj zarówno oceny jedno- jak i wieloetapowe. Technicznie ich obsługa w systemie we wnioskach NCBR sprowadza się jednak do uzupełnienia dedykowanego formularza oceny lub podsumowania oceny, która została wykonana poza systemem (np. na posiedzeniu zespołu), lub użycia modułu recenzji – który jest najczęściej stosowany – umożliwia on wysyłanie indywidualnym ekspertom lub członkom paneli oceniających, zaproszenia do wykonania recenzji wniosku, wykonanie tej recenzji on-line i przesłanie oceny wniosku pracownikom NCBR.

W zależności od zapisów regulaminowych konkursu i potrzeb NCBR pojawiają się w systemie następujące typy ocen obsługiwane na jeden z wyżej wymienionych sposobów:

- Ocena zgodności w formie formularza do uzupełnienia – wstępna weryfikacja czy temat projektu jest zgodny z tematyką programu, z którego finansowany jest konkurs;
- Recenzje ekspertów w module recenzji;
- Ocena prepanelowa w module recenzji;
- Ocena ekspercka/panelowa/recenzje II stopnia – w postaci podsumowania wyników panelu w postaci formularza uzupełnianego przez sekretarza panelu ekspertów lub recenzji wykonywanych w module recenzji przez ekspertów;
- Ocena zespołu interdyscyplinarnego.

W NCBR do wyliczenia ostatecznej oceny wniosków stosuje się mechanizm recenzji. Stanowi to kontrast do procesów MEiN czy NCN, gdzie recenzje ekspertów są tylko częściowo wykorzystywane w procesach obsługi. W procesie oceny wniosków NCBR nie ma referentów wniosków oraz stałych (lub wybieranych na pewien dłuższy okres/kadencję) składów zespołów oceniających wnioski. NCBR ma możliwość przypisania jako osoby oceniającej dowolnej osoby z bazy systemu (lub spoza niej), którą wewnętrznie identyfikuje jako eksperta w danej dziedzinie. Lista dostępnych ekspertów nie jest zatem ograniczana do listy członków zespołu zdefiniowanego w systemie – jak ma to miejsce w MEiN i NCN.

W procesie obsługi niektórych wniosków NCBR, po ocenie merytorycznej, występuje etap negocjacji, co wyróżnia moduł NCBR od pozostałych. Z wnioskodawcami, których wniosek został zarekomendowany do finansowania, mogą zostać przeprowadzone negocjacje, w wyniku których ustalana jest ostateczna kwota dofinansowania. Formalnie negocjacje mogą zostać przeprowadzone w przypadku zastrzeżeń dotyczących zasadności kosztów realizacji projektu lub przeprowadzonej analizy pod kątem kwalifikowalności kosztów, przyporządkowania kosztów do właściwej kategorii oraz zadań do właściwej fazy i rodzaju badań itp. Właściwe negocjacje są prowadzone poza systemem informatycznym, jednak ich efekty można wprowadzić bezpośrednio w systemie OSF. Skorygowane dane są potem brane pod uwagę przy generowaniu umowy o finansowaniu. Negocjacje i wprowadzanie korekty we wniosku jest jednak krokiem opcjonalnym, można go pominąć i od razu wydać decyzję o finansowaniu.

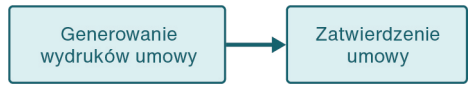
Decyzje o ostatecznym wyniku oceny merytorycznej i przyznaniu finansowania w NCBR są ustalane na podstawie list rankingowych. System OSF w przypadku wniosków NCBR umożliwia wyłącznie eksport zestawienia stanowiącego podstawę do jej przygotowania. System wylicza punktację wynikającą z oceny merytorycznej wniosku w ramach eksportu. Formalne przygotowanie ostatecznej listy rankingowej i wyznaczenie wniosków do finansowania ma miejsce poza systemem. Pracownik NCBR indywidualnie wprowadza potem decyzję w systemie, dla każdego wniosku z osobna.

Wprowadzone do systemu decyzje muszą zostać zatwierdzone. Podobnie jak przy ocenie formalnej, zatwierdzenia oceny merytorycznej może dokonać tylko zaawansowany opiekun wniosku, a operacja jest nieodwracalna. Każdorazowo jednak przed zatwierdzeniem oceny merytorycznej (pierwszego lub każdego kolejnego stopnia), wszystkie wnioski muszą być odpowiednio przygotowane. Jeśli to ostatni stopień oceny – powinny mieć wprowadzoną decyzję, która z momentem zatwierdzenia staje się w systemie uznana za ostateczną. Zatwierdzenie oceny merytorycznej kończy proces dla większości wniosków NCBR.

2.3. Obsługa finansowania

Proces obsługi finansowania rozpoczyna się w momencie zakończenia etapu oceny merytorycznej dla wniosków, które decyzją danej instytucji otrzymały finansowanie. Od tego momentu możliwe jest podpisanie umowy między badaczem lub podmiotem realizującym projekt badawczy a instytucją odpowiedzialną za finansowanie, zaś przedmiotem finansowania jest określony we wniosku projekt. Wsparcie systemu OSF dla tego etapu jest zróżnicowane, w zależności od modułu.

OBSŁUGA FINANSOWANIA W NCBR

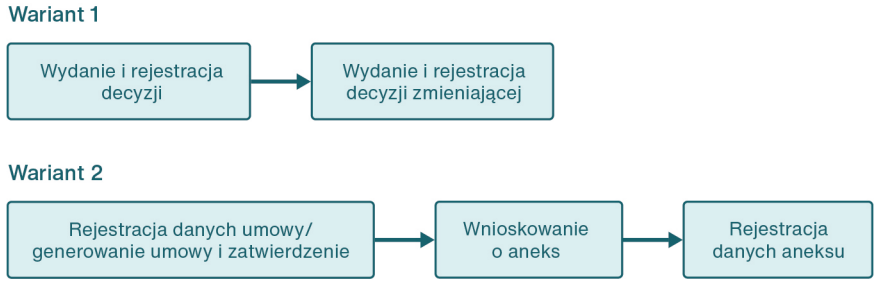


Ilustracja 2.6. Etapy procesu obsługi finansowania w Narodowym Centrum Badań i Rozwoju

Dla większości wniosków NCBR proces obsługi wniosku kończy się w momencie zatwierdzenia oceny merytorycznej (por. Ilustracja 2.6). Dla części z nich możliwe jest jednak wygenerowanie wydruków umowy w postaci plików PDF, a dla jeszcze mniejszej części – zatwierdzenie umowy wraz z rejestracją daty jej podpisania. Procesy NCBR występujące w OSF nie przewidują obsługi dalszych etapów życia projektu.

OBSŁUGA FINANSOWANIA W MEIN

W ramach modułu MEiN obsługa finansowania różni się w zależności od rodzaju wniosku. Dla większości rodzajów wniosków obsługa finansowania polega na obsłudze umów i aneksów, jednakże nie dla wszystkich wniosków występuje umowa (por. Ilustracja 2.7).



Ilustracja 2.7. Etapy procesu obsługi finansowania w Ministerstwie Edukacji i Nauki – warianty

Dla części wniosków proces w systemie OSF nie kończy się umową, bądź umowa nie jest rejestrowana w systemie. Dla przykładu we wniosku o przyznanie środków finansowych na utrzymanie aparatury naukowo-badawczej, stanowiska badawczego lub na utrzymanie specjalnej infrastruktury informatycznej umowa nie występuje, zamiast tego wydawana jest decyzja. Jeśli zachodzi konieczność zmiany zapisów określonych w decyzji, wydawana jest tzw. decyzja zmieniająca.

Dla wniosków MEiN objętych standardem w systemie OSF rejestrowane są dane umowy w zakresie daty podpisania umowy oraz wypełniane są dodatkowe sekcje, wykorzystywane w kolejnym etapie, tj. przy raportach. Po podpisaniu umowy w systemie wprowadzana jest data podpisania umowy. Jedynym wnioskiem realizowanym w ra-

mach MEiN, w którym generowana (a nie tylko rejestrowana w systemie) jest umowa, jest wniosek w ramach programu „Doktorat wdrożeniowy". Umowę można również rozwiązać, wprowadzając w systemie datę rozwiązania umowy.

Podobny proces jak przy umowie występuje dla aneksów wprowadzanych dla wniosków objętych standaryzacją. Tam również w systemie rejestrowany jest aneks, występują dodatkowe sekcje (takie same jak przy umowie), następnie wprowadza się datę podpisania aneksu.

Dla wniosków w ramach programu „Doktorat wdrożeniowy" dodatkowo występuje możliwość złożenia wniosku o aneks, a następnie rejestracji aneksu. W ramach wniosku o aneks można zgłosić zmiany do pierwotnie zgłoszonego wniosku np. dodać kolejnego uczestnika programu. Taka funkcjonalność występuje jedynie w tym typie wniosku.

OBSŁUGA FINANSOWANIA W NCN

Ilustracja 2.8. Etapy procesu obsługi finansowania w Narodowym Centrum Nauki

W module NCN dla wniosków z pozytywną decyzją o finansowaniu możliwe jest przygotowywanie umów za pośrednictwem systemu. Formularze umów w nowych edycjach konkursów są uzupełniane automatycznie na podstawie danych z wniosków, z możliwością wprowadzania modyfikacji zarówno przez pracownika NCN, jak i redaktora wniosku – w wyznaczonym zakresie. W tym celu projekt umowy przekazywany jest za pośrednictwem systemu pomiędzy stronami, do momentu zaakceptowania przez pracownika NCN finalnej wersji umowy, gotowej do podpisania. Po zatwierdzeniu danych umowy, generowany jest dokument umowy, zaś po jej podpisaniu, fakt ten rejestrowany jest w systemie (por. Ilustracja 2.8).

System umożliwia dodatkowo odnotowanie faktu wstrzymania finansowania oraz wycofania wstrzymania finansowania, a także odstąpienia od umowy oraz wycofania odstąpienia.

W trakcie trwania umowy możliwe jest także zgłoszenie w systemie OSF zmiany bądź aktualizacji danych umowy oraz przygotowanie aneksu do umowy i porozumienia zmieniającego.

2.4. Rozliczanie finansowania

Funkcjonalności związane z rozliczaniem projektów nie są dostępne w OSF dla wniosków NCBR. W pozostałych modułach możliwe jest składanie raportów związanych z realizacją projektu badawczego objętego finansowaniem, co ma na celu kontrolę realizacji projektu oraz prawidłowości wydatkowania środków pochodzących z dofinansowania. Zarówno w przypadku wniosków Narodowego Centrum Nauki, jak i wniosków ministerialnych, wnioskodawcy są zobligowani do składania następujących rodzajów raportów:

- rocznych (po każdym roku trwania projektu);
- końcowych (najczęściej po ostatnim roku lub w ostatnim roku realizacji projektu).

System umożliwia przygotowanie projektu raportu w taki sposób, że część danych pobierana jest automatycznie z zapisanych w systemie wniosków oraz umów. Pozostałe dane wnioskodawca uzupełnia informacjami wynikającymi z rezultatów, jakie zostały osiągnięte w ramach projektu badawczego.

ROZLICZANIE FINANSOWANIA W MEIN

Wszystkie raporty w systemie przedkładane w ramach MEiN składają się z następujących etapów: wypełnienie i wysłanie formularza raportu, ocena formalna oraz ocena merytoryczna. Podobnie jak dla wniosków podczas oceny formalnej można odesłać raport do korekty do redaktora. We wnioskach objętych standardem podczas oceny formalnej raportu wnioski są kierowane do oceny finansowej (dotyczy raportów końcowych, w przypadku raportów rocznych w zależności od decyzji opiekuna merytorycznego wniosku/raportu). We wnioskach nieobjętych standardem ten krok nie obowiązuje. Ocena merytoryczna raportów dla wniosków objętych standardem składa się z opinii referenta, która musi zostać zaakceptowana przez przewodniczącego zespołu specjalistycznego, a następnie opiekun wniosku udostępnia ocenę i ustawia status raportu.

We wniosku realizowanym w ramach programu „Doktoraty wdrożeniowe" ocena merytoryczna raportu składa się z oceny referenta, oceny końcowej, udostępnienia oceny a także z decyzji Ministra. Raport roczny można odesłać do uzupełnienia do redaktora po ocenie merytorycznej. Kryteria oceny są zmieniane dla każdego kolejnego konkursu.

ROZLICZANIE FINANSOWANIA W NCN

Raporty składane do NCN działają w systemie podobnie jak w przypadku raportów MEiN. Obejmują przygotowanie raportu, ocenę formalną oraz merytoryczną. W przypadku raportu końcowego dokonywana jest weryfikacja, czy finanse przyznane w ramach grantu bądź stypendium zostały właściwie spożytkowane. Projekt zakończony, oceniony pozytywnie, zostaje oznaczony jako rozliczony.

ROZDZIAŁ 3

PROCES TECHNOLOGICZNY

Przemysław Zydroń

W historii struktury i kształtu zespołu realizującego projekt OSF można wydzielić trzy etapy dojrzałości. Pierwszy z nich rozpoczął się w 2005 roku, czyli w momencie powołania projektu do życia. Zespół był wówczas niewielki, początkowo składał się z trzech osób. Jego struktura była płaska, charakterystyczna bardziej dla startupu, w którym kierownik projektu pełni również funkcję analityka i testera. Poszczególni członkowie zespołu nie mieli ściśle przypisanych ról w projekcie. Początkowo zakładano, że projekt będzie trwał około 18 miesięcy. W ciągu następnych pięciu lat, w związku ze wzrostem zapotrzebowania na kolejne moduły i funkcje, liczba członków zespołu zwiększyła się do 10.

W okolicach roku 2010, wraz z pojawianiem się Narodowego Centrum Badań i Rozwoju (NCBR) jako trzeciego interesariusza obsługiwanego przez system, pojawił się pomysł stworzenia osobnych zespołów przypisanych każdej z obsługiwanych instytucji. I tak, pod koniec 2012 roku w projekcie funkcjonowały trzy zespoły. Każdy dedykowany oddzielnemu podmiotowi. Te podmioty to:

- Ministerstwo Nauki i Szkolnictwa Wyższego;
- Narodowe Centrum Nauki;
- Narodowe Centrum Badań i Rozwoju.

Co bardzo istotne, wraz z powołaniem trzech zespołów nie zdecydowano się na zmiany dotyczące architektury systemu. Czyli trzy zespoły pracowały nad rozwojem jednej aplikacji. Pojawił się problem konieczności koordynacji prac pomiędzy poszczególnymi zespołami.

Warto zauważyć, że zespoły były nadal multidyscyplinarne, tzn. w ich skład wchodzili analitycy, programiści i testerzy. A role były zazwyczaj łączone.

Odejście od multidyscyplinarności zespołów nastąpiło dopiero w 2016 roku. Impulsem do wprowadzania tych zmian było powołanie projektu Zintegrowanego Systemu Usług dla Nauki etap I (ZSUN I). Wraz z rozpoczęciem realizacji nowego projektu i wzrostem poziomu finansowania systemu, pojawiła się potrzeba zwiększenia jego zasobów kadrowych oraz zmian w strukturze zespołów.

Nadal pozostały trzy zespoły programistyczne. Ale w ich skład wchodzili teraz wyłącznie programiści. Każdy z zespołów programistycznych dedykowany był jednej z trzech obsługiwanych instytucji. Oprócz tego powołano do życia zespół analityczny oraz zespół testów. Jako szósty, bez sformalizowanej struktury, wyłonił się zespół obsługujący platformę, czyli część wspólną realizującą wszystkie wspólne, podstawowe mechanizmy systemu. W jego skład wchodzili architekci, administratorzy i specjaliści od graficznego interfejsu użytkownika (ang. *GUI*). Wymogi projektu, realizowanego w ramach dotacji unijnej, wymusiły stosowanie procesu wytwórczego opartego o elementy metodyki PRINCE2. Było to widoczne zwłaszcza w odniesieniu do sposobu budowy harmonogramu dla zadań realizowanych na rzecz Ministerstwa Nauki i Szkolnictwa Wyższego.

Uruchomienie projektu ZSUN I wiązało się z przejściem na nowsze technologie ale, co ważne, nadal nie zdecydowano się na poważne zmiany w architekturze systemu. Cały zespół pracował nad jedną, stanowiącą monolit, aplikacją.

Dopiero na przełomie lat 2019 i 2020 rozpoczęto prace nad zmianą architektury i przejściem na model, w którym dokonano podziału na osobne, dedykowane każdej z obsługiwanych instytucji podsystemy oraz dostarczającą im uniwersalne usługi część wspólną. Zmiany w architekturze wiązały się także z pewnymi zmianami w procesie wytwórczym.

3.1. Proces analizy wymagań

3.1.1. Analiza wstępna wymagań

Proces analizy wymagań rozpoczyna się od wpłynięcia do zespołu analitycznego wstępnych założeń proponowanej zmiany. Są one przekazywane na kilka sposobów:

- Drogą mailową;
- Jako wynik spotkań roboczych z przedstawicielami poszczególnych instytucji; w takich wypadkach najczęściejsporządzana jest notatka i to ona jest podstawą procedowania zmian;
- Tworzone jest zgłoszenie w dedykowanym do obsługi zadań i zgłoszeń narzędziu Redmine[14].

Następnie dokonywana jest analiza wstępna. Jest to podproces, czyli pierwszy krok będący składnikiem procesu analizy zmiany. W ramach tego podprocesu można wskazać kilka czynności. A że jest to tzw. podproces ad-hoc, czynności w nim mogą być wykonywane w dowolnej kolejności oraz dowolną liczbę razy. Wszystko to zależy od

[14] Narzędzie dozarządzania projektami i zadaniami, redmine.org (dostęp: 17.08.2021)

aktualnych potrzeb. W zależności od rozmiaru i stopnia złożoności zmiany, analiza wstępna może przyjąć różne formy. W trakcie jej trwania wymagane jest stworzenie zgłoszenia w Redmine. Jeśli zgłoszenie takie nie zostanie stworzone, zmiana nie wyjdzie poza analizę wstępną i nie będzie w przyszłości realizowana. W ramach podprocesu analizy wstępnej można wskazać następujące kroki:

- Analiza wstępna materiałów;
- Przygotowanie materiałów roboczych;
- Konsultacje z architektami i programistami;
- Uzyskanie i potwierdzanie wymagań;
- Zainicjowanie i dokumentowanie wymagań;
- Zainicjowanie zadania w harmonogramie;
- Podział zadania na etapy.

Analiza wstępna materiałów: obejmuje dokładne zapoznanie się z materiałami oraz ich weryfikację pod względem kompletności i spójności wymagań.

Przygotowanie materiałów roboczych: krok obejmuje, w zależności od potrzeby, stworzenie makiet interfejsu użytkownika oraz identyfikacji i opisu najważniejszych algorytmów. Elementy wytworzone w ramach tego kroku są wykorzystywane w kroku „Zainicjowanie i dokumentowanie wymagań".

Konsultacje z architektami i programistami: analityk potwierdza techniczną wykonalność wymagań, ze szczególnym uwzględnieniem najbardziej złożonych elementów.

Uzyskiwanie i potwierdzanie wymagań: w tej czynności instytucja proszona jest o odpowiedź na pytania analityka oraz, w razie konieczności, dostarczenie materiałów uzupełniających, odpowiedź na dodatkowe pytania lub wyjaśnienie wątpliwości bezpośrednio na spotkaniu (tzw. warsztacie wymagań).

Zainicjowanie i dokumentowanie wymagań: podproces obejmuje utworzenie dokumentacji analitycznej na podstawie konkretnych wymagań. W tym kroku wykorzystywane są materiały wytworzone w ramach podprocesu „Przygotowanie materiałów roboczych".

Zainicjowanie zadania w harmonogramie: każda zmiana, która wymaga zaangażowania zespołu programistycznego, a nie jest błędem, jest umieszczana w harmonogramie jako osobne zadanie. Do umieszczenia zadania w harmonogramie niezbędne jest utworzenie odpowiadającego mu zgłoszenia w systemie śledzenia zgłoszeń Redmine.

Podział zadania na etapy: złożone zadania są dzielone na mniejsze części, tzw. etapy. Celem tego kroku jest jak najszybsze dostarczenie funkcji produktu zamawiającemu oraz otrzymanie informacji zwrotnej dotyczącej prawidłowej interpretacji wymagań. Dodatkowo rozbicie na części składowe służy zmniejszeniu poziomu złożoności każdego z etapów.

3.1.2. Jeden punkt zgłoszeń

Bardzo ważnym aspektem procesu obsługi zgłoszeń w obrębie systemu OSF jest istnienie jednego i tylko jednego punktu zgłoszeń. Funkcję tego punktu pełni aplikacja Redmine. Z narzędzia korzystają wszyscy interesariusze systemu, wewnętrzni i zewnętrzni. Jego podstawową zaletą jest fakt, że już na wstępnym etapie wszystkie zgłoszone zagadnienia trafiają w jedno miejsce. I w jednym miejscu są obsługiwane. Zasada ta została zaimplementowana na podstawie zaleceń biblioteki ITIL.

3.1.3. Analiza pełna etapu

Analiza pełna etapu, w przeciwieństwie do analizy wstępnej, jest zdecydowanie bardziej uporządkowanym procesem. Poszczególne kroki analizy pełnej następują po sobie w ściśle określonej kolejności. Proces rozpoczyna się, gdy wszystkie lub prawie wszystkie wątpliwości wykryte na etapie analizy wstępnej są wyjaśnione z zamawiającym. Co ważne, decyzja o zakończeniu analizy wstępnej podejmowana jest przez analityka zajmującego się zagadnieniem wg jego uznania. Pierwszym krokiem procesu jest uzupełnienie i zakończenie prac nad dokumentacją wytworzoną w ramach analizy wstępnej. Powstaje dokument analityczny zgodny z szablonem obowiązującym w laboratorium. W jego skład wchodzą:

- Diagramy przypadków użycia;
- Opis przypadków użycia;
- Projekt elementów graficznego interfejsu użytkownika;
- Opis reguł biznesowych, walidacyjnych oraz bardziej skompilowanych algorytmów;
- W uzasadnionych przypadkach dodatkowe projekty techniczne planowanych rozwiązań.

Na tym etapie cała komunikacja z zamawiającym jest prowadzona przy użyciu aplikacji Redmine. Z reguły, by uniknąć nadmiernej ilości przebiegów, pytania są grupowane.

Gotowe materiały umieszczane są w używanym przez zespół repozytorium. Przed rozpoczęciem implementacji niezbędna jest akceptacja spisanych wymagań przez zamawiającego. W przypadku bardziej złożonych funkcji organizowane jest spotkanie, na którym dokumentacja jest prezentowana. Na tym etapie zamawiający może zgłosić jeszcze swoje uwagi, które są uwzględniane. Czynność ta często ma charakter iteracyjny. Ostateczne zaakceptowanie dokumentacji powoduje przystąpienie do implementacji. Rozpoczęcie wytwarzania oprogramowania kończy proces analizy, choć nie kończy procesu jako całości.

Zdarzeniem inicjującym implementację jest tzw. *kick-off*, na którym analityk odpowiedzialny za moduł prezentuje założenia dotyczące nowej lub modyfikowanej funkcji. Prezentowane są najważniejsze elementy dokumentacji tj. przypadki użycia, projekt interfejsu użytkownika, diagramy przepływu oraz ważniejsze algorytmy.

W *kick-off* udział biorą wszystkie osoby, które uczestniczyć będą w dalszej części procesu: analityk odpowiedzialny za wykonanie dokumentacji, programiści oraz testerzy. Na spotkanie zapraszany jest również przedstawiciel instytucji.

Celem spotkania jest przede wszystkim zaznajomienie zespołu realizującego funkcjonalność z wymaganiami. Omawia się na nim główne przypadki użycia, procesy, bardziej skomplikowane walidacje, wybrane aspekty techniczne implementacji. Bardzo ważnym elementem tego spotkania jest to, że podjęte w jego trakcie ustalenia mogą nadal powodować zmiany w produkcie docelowym. Oznacza to, że obecność klienta jest kluczowa dla zatwierdzenia wprowadzanych zmian. Zgłoszone w ten sposób modyfikacje mają również odzwierciedlenie w dokumentacji. Nowa wersja dokumentacji również wymaga akceptacji zamawiającego, ale nie wpływa to na wstrzymanie implementacji.

Proces implementacji jest poprzedzony wyceną pracochłonności. W zależności od zespołu dokonuje jej lider lub programiści. W przypadku wyceny dokonywanej przez więcej niż jedną osobę wykorzystywany jest tzw. *planning poker*.

3.2. Proces implementacji

Punktem wyjścia dla procesu implementacji jest zatwierdzenie wymagań przez zamawiającego. Zdarzeniem rozpoczynającym prace zespołu programistycznego jest spotkanie inicjujące, czyli *kick-off*. Biorą w nim udział wszystkie osoby, które będą uczestniczyły w dalszej części procesu: analityk odpowiedzialny za wykonanie dokumentacji, programiści i testerzy. Na spotkanie zapraszany jest również przedstawiciel reprezentujący klienta. W ramach spotkania omawiany jest pełen zakres planowanej funkcjonalności.

Po zakończeniu *kick-off* następuje robocze spotkanie programistów wyznaczonych do implementacji. Dzielą oni funkcjonalność na mniejsze zadania i określają ich pracochłonność. Powstała w ten sposób wycena zadań cząstkowych służy do stworzenia ostatecznej wyceny całej funkcjonalności oraz podziału pracy na etapy. Do każdego etapu są przydzielane zadania, które zespół programistyczny jest w stanie wykonać w ciągu jednego tygodnia roboczego. I tak funkcjonalność oceniona na 15 dni roboczych zespołu, będzie realizowana w trzech etapach.

W czasie podziału pracy na etapy podejmowana jest decyzja, czy poszczególne etapy będą wdrażane wraz z zakończeniem prac nad całą funkcjonalnością, czy też wdrażane będą osobno. Decyzja każdorazowo konsultowana jest z zamawiającym i dopasowana do jego potrzeb. W procesie implementacji realizowane są główne zadania programistyczne. Dla każdego z nich programiści tworzą swoje podzadania, w ramach których są realizowane właściwe prace. Dodatkowo tworzona jest gałąź (ang. *branch*) w repozytorium kodu źródłowego, do której trafiają wszystkie nowe i modyfikowane fragmenty kodu odpowiadające za realizację tworzonej funkcjonalności.

Każde wykonane podzadanie programistyczne trafia do oceny, za którą odpowiada inny programista. Jest to tzw. przegląd (ang. *review*) kodu. W ramach przeglądu oceniana jest jakość kodu, jego zgodność z przyjętymi w zespole standardami. W ramach przeglądu nie jest sprawdzana zgodność kodu z wymaganiami. Dopiero pozytywna ocena kodu pozwala uznać prace programistyczne za ukończone oraz przekazać je do testów.

Jeśli dane zadanie ma wysoki priorytet, zaraz po ukończeniu kierowane jest do testów. W przypadku mniej pilnych funkcjonalności testy następują dopiero po zakończeniu prac nad danym etapem lub po zakończeniu prac nad całą funkcjonalnością. Po wykonaniu niezbędnych testów podzadanie programistyczne może być:

- Odesłane do poprawy;
- Odesłane do analityka;
- Uznane za wykonane.

3.2.1. Odesłane do poprawy

W przypadku wykrycia błędu lub usterki tester zwraca zadanie programiście, który je realizował, zlecając tym samym naprawienie błędu.

3.2.2. Odesłane do analityka

W przypadku wątpliwości dotyczącej zgodności zaimplementowanej funkcji z zawartymi w dokumentacji założeniami, tester może zadać pytanie analitykowi celem wyjaśnienia problemu. W zależności od odpowiedzi zadanie programistyczne może zostać odesłane do poprawy lub uznane za wykonane poprawnie.

3.2.3. Uznane za wykonane

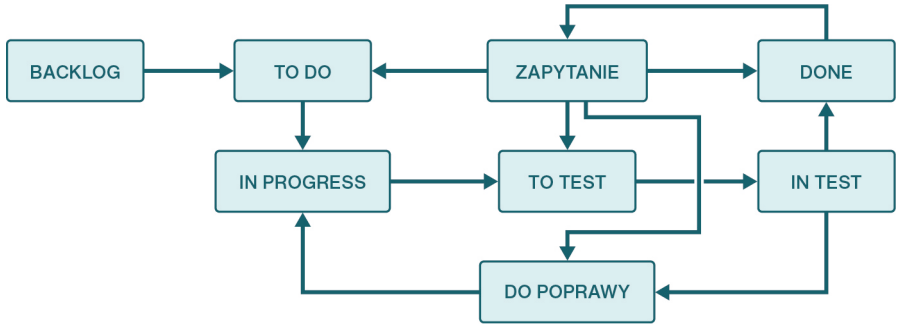
Jeśli w wyniku testów tester nie zidentyfikuje żadnego błędu lub usterki podzadanie zostaje uznane za wykonane i jest mu nadawany odpowiedni status (ang. *Done*).

Po zakończeniu pracy nad całą funkcjonalnością, kod źródłowy scalany jest (ang. *merge*) z podstawową gałęzią projektu i kierowany do testów wewnętrznych. Tam aplikacja poddawana jest testom integracyjnym. Głównym celem testów jest sprawdzenie zgodności nowych funkcji aplikacji z dokumentacją. Na tym etapie wykrywana jest zwykle największa liczba błędów.

Gdy testy integracyjne zostaną zakończone, nowe elementy aplikacji osiągną odpowiednią dojrzałość, a w systemie nie będą pojawiały się błędy krytyczne lub blokujące, funkcjonalność zostaje przekazana do testów akceptacyjnych. Ich celem jest ostateczne potwierdzenie zgodności aplikacji z wymaganiami klienta. Na tym etapie zamawiający, mając dostęp do gotowej wersji systemu, ma możliwość zgłaszania swoich zastrze-

żeń. Pozwala to na zwiększenie wartości oprogramowania dla klienta. Zgłoszone w ten sposób zmiany ponownie przechodzą cały proces realizacji implementacji, w tym także testy akceptacyjne. W przypadku zaakceptowania przez zamawiającego całej funkcjonalności jest ona kierowana do wdrożenia.

Wdrożenie produkcyjne poprzedzone jest testami typu *release*. Środowisko testów *release* ma w jak największym stopniu odwzorowywać środowisko produkcyjne. Celem tych testów jest sprawdzenie czy przygotowany do wdrożenia pakiet oprogramowania jest kompletny. Ten etap służy znalezieniu ewentualnych braków lub błędów technicznych. Nie są już za to sprawdzane wymagania biznesowe klienta. A co za tym idzie zmiany wprowadzone na tym etapie nie przechodzą przez testy akceptacyjne klienta. Zakończenie prac nad poprawkami powstałymi na podstawie testów release kończy pracę zespołu programistycznego nad zmianą. Samo wdrożenie na produkcję jest realizowane przez administratorów. Kompletna lista stanów podzadania programistycznego zaprezentowana jest na Ilustracji 3.1.



Ilustracja 3.1. Przebieg podzadania programistycznego

3.3. Proces testów

Sygnałem do rozpoczęcia prac przez zespół testerski nad daną funkcjonalnością jest oznaczenie powiązanego z nią zadania jako wykonanego przez zespół programistyczny.

Wyznaczony tester rozpoczyna pracę nad zagadnieniem od zapoznania się z dokumentacją. Następnie na jej podstawie przygotowuje plan testów. Plan testów służy logicznemu podziałowi funkcjonalności na mniejsze, powiązane ze sobą elementy. Dodatkowo na podstawie planu tworzone są podzadania testowe, w ramach których realizowane są testy funkcjonalne. Z jednej strony pozwala to osobie realizującej testy skupić się w danym momencie tylko na części funkcjonalności. A dodatkowo ułatwia w razie potrzeby zwiększenie liczby osób zaangażowanych w testy.

Same testy polegają na weryfikacji zgodności funkcjonalności z dokumentacją na podstawie, której została stworzona. Weryfikowane są zarówno wymagania funkcjonalne jak i niefunkcjonalne. W ramach tego etapu prowadzone są również testy eksploracyjne w oparciu o wiedzę domenową testera.

W przypadku wykrycia nieprawidłowego działania funkcjonalności, tworzone jest zadanie podrzędne do podzadania testowego zawierające opis błędu. W skład opisu wchodzi:

- Opis błędu;
- Opis sposobu reprodukcji (odtworzenia) błędu;
- Adres URL miejsca, gdzie błąd został wywołany;
- Załączniki zawierające zrzuty ekranu, film z nagraniem momentu wystąpienia oraz inne materiały pomocne w identyfikacji przyczyn błędu.

Wymienione elementy zgłoszenia błędu służą następnie programistom do identyfikacji przyczyn wstąpienia błędu, a następnie jego poprawy. Gdy poprawka zostaje wykonana, tester dokonuje retestu funkcjonalności w miejscu, gdzie wystąpił błąd. W przypadku, gdy błąd nadal występuje, zgłoszenie zostaje odesłane do poprawy. Jeśli błąd został poprawiony, tester uznaje zgłoszenie błędu za wykonane. W szczególnych przypadkach, gdy poprawka błędu generuje nowy błąd, tworzone jest nowe zgłoszenie błędu. Proces obsługi błędu zgłoszonego na tym etapie nie różni się niczym od wcześniej opisanego.

Jeśli tester napotka na problem, którego nie jest w stanie jednoznacznie zidentyfikować jako błąd, ma możliwość poproszenia analityka o wsparcie. W takim przypadku tworzone jest zagadnienie analityczne zawierające następujące elementy:

- Opis problemu;
- Opis fragmentu dokumentacji analitycznej, której problem dotyczy;
- Załączniki zawierające zrzuty ekranu, film z nagraniem momentu wystąpienia oraz inne materiały pomocne w identyfikacji przyczyn problemu.

Na podstawie przesłanych materiałów oraz swojej wiedzy domenowej analityk może wykonać jedną z trzech rzeczy:

- Potwierdzić, że zgłoszony przez testera problem jest błędem. W takiej sytuacji analityk zmienia rodzaj zagadnienia na błąd i przekazuje do programistów;
- Zidentyfikować, że problem powoduje błąd w dokumentacji. W takim wypadku analityk poprawia błąd w opisie funkcjonalności i odsyła zgłoszenie do testera;
- Uznaje, że system działa zgodnie z dokumentacją. W takim przypadku zagadnienie dot. prośby o wsparcie jest zamykane przez testera.

Proces testowania funkcjonalności jest realizowany iteracyjnie. Po zakończeniu każdej iteracji tworzony jest zestaw błędów, który trafia do poprawy do zespołu programistycznego. Kolejna tura testów zaczyna się po naprawieniu błędów przez zespół programistyczny.

Na zakończenie każdej iteracji tworzone jest podsumowanie, dla każdego zadania testowego stworzonego na podstawie planu testów. Dodatkowo tworzone jest podsumowanie zbiorcze całej rundy testów.

Testy całej funkcjonalności kończą się po naprawieniu wszystkich lub prawie wszystkich wykrytych błędów. Na zakończenie testów tworzone jest podsumowanie. Zawiera informacje, co udało się przetestować. Co jeszcze nie zostało naprawione oraz co będzie testowane w późniejszym terminie, w ramach wydzielonego osobnego zgłoszenia.

Funkcjonalność, dla której zakończone są testy wewnętrzne, kierowana jest na testy akceptacyjne do zamawiającego.

3.3.1. Testy release

Po zaakceptowaniu funkcjonalności przez zamawiającego, a przed jej wdrożeniem wykonywane są tzw. testy *release*. Ich celem jest sprawdzenie czy pakiet zmian przygotowanych do wdrożenia jest poprawny pod względem technicznym w środowisku jak najbardziej zbliżonym do produkcyjnego. W ramach testów weryfikowane jest poprawne działanie systemu, a nie zgodność systemu z dokumentacją. W związku z powyższym w głównej mierze są one oparte o testy eksploracyjne.

Sam proces zaczyna się od zbudowania i wdrożenia na serwer *release* wersji aplikacji przeznaczonej do wdrożenia. Tworzone jest zadanie główne, w ramach którego będą realizowane testy. Do zadania głównego na podstawie wpisów w *changelogu* (czyli liście zmian w aplikacji) tworzone są podzagadnienia. Osobno dla każdej z wdrażanych funkcjonalności. Sam przebieg testów jest analogiczny do przebiegu testów wewnętrznych.

Na zakończenie testów tworzone jest podsumowanie każdego z podzadań jak i całego *release*. Dopiero po zakończeniu testów *release* możliwe jest wdrożenie nowej wersji aplikacji.

ROZDZIAŁ 4

ARCHITEKTURA SYSTEMU

Adam Bochenek
Jacek Bieliński

W rozdziale pierwszym wspomnieliśmy, że pierwsze wiersze kodu źródłowego systemu OSF pojawiły się 29 czerwca 2005 roku. Ale tworzenie każdego systemu informatycznego (czy nawet pojedynczego jego modułu) nie może i nie sprowadza się wyłącznie do kodowania. Pisanie kodu jest tylko jednym z etapów pracy. W poprawnie prowadzonych projektach kodowanie powinno być poprzedzone rzetelnie wykonaną analizą i pracami projektowymi. Ale i to nie wystarczy. Wszystkie te elementy muszą wpisywać się w przemyślaną i dobrze skonstruowaną architekturę systemu.

Czym jest architektura?

Definicja architektury oprogramowania ewoluowała w ciągu ostatnich kilkudziesięciu lat. Pod koniec lat sześćdziesiątych architektura oznaczała przede wszystkim strukturę sprzętu i oprogramowania [6]. System informatyczny, jako całość, postrzegany był jako konstrukcja składająca się ze współpracujących ze sobą składników. Z czasem znaczenie pojęcia „architektura" zaczęło się rozszerzać. Według wielce zasłużonych dla inżynierii oprogramowania Grady'ego Boocha, Jamesa Rumbaugha i Ivara Jacobsona architektura systemu to zbiór decyzji dotyczących organizacji systemu, wyboru elementów strukturalnych, sposobu kooperacji komponentów, wyboru stylu całej konstrukcji czy sposobu tworzenia interfejsów [5].

Jeszcze dalej w swoich rozważaniach na temat architektury idzie Robert C. Martin. Uważa on, że architektura nie odpowiada wyłącznie za ustalony na początku projektu kształt systemu, ale wpływa także na jego przyszłe losy. Umożliwia i ułatwia dalszy rozwój, konserwację, możliwość modyfikacji. Czyli architektura ma wpływ na prace w całym cyklu życia aplikacji, łącznie z jego rozbudową, gdy zaistnieje taka potrzeba. Z tego punktu widzenia przygotowanie działającego systemu jest tylko jednym z celów [18]. Równie istotnym jest możliwość rozwoju, zmian, zapewnienie niezależności od sprzętu czy łatwość wdrażania.

Niniejszy rozdział opisuje system OSF z perspektywy jego architektury, biorąc pod uwagę wszystkie aspekty, które termin ten obejmuje.

4.1. Pierwsze decyzje architektoniczne

Rozpoczynając pracę nad nowym systemem informatycznym należy podjąć wiele decyzji dotyczących architektury i narzędzi, za pomocą których będzie on tworzony. Dokonując wyboru trzeba wziąć pod uwagę wiele czynników. Wśród nich są dominujące w danym okresie standardy, technologie, perspektywy ich rozwoju, trendy. Równie ważne są preferencje, możliwości i doświadczenie zespołu czy organizacji, która system ma tworzyć. Nie wolno też zapomnieć o tzw. wymaganiach pozafunkcjonalnych, czyli takich, które wiążą się z wydajnością systemu, liczbą obsługiwanych użytkowników, odpornością na awarie, możliwością skalowania systemu itp. Spójrzmy z tej właśnie perspektywy na system OSF i decyzje podjęte na przełomie lat 2004 i 2005.

4.1.1. Dominujące na rynku technologie

W okolicach roku 2004 na rynku technologii służących do tworzenia aplikacji internetowych istniało trzech poważnych graczy: PHP, Java (J2EE) oraz ASP/ASP.NET. Oczywiście wszystkich dostępnych narzędzi, języków programowania czy szkieletów aplikacji (ang. *framework*) było znacznie więcej, ale te trzy zasługiwały na uwagę i były na etapie decyzji dotyczących systemu OSF wzięte pod uwagę.

PHP

Historia tego interpretowanego języka programowania sięga roku 1994. Od samego początku najczęściej wykorzystywany był on do dynamicznego generowania stron internetowych po stronie serwera. Niewielki na starcie projekt zyskał kilka lat później ogromną popularność. Implementacja PHP działająca w połączeniu z systemem Linux, serwerem WWW Apache i bazą danych MySQL (w skrócie LAMP) stała się jednym z dominujących standardów w świecie aplikacji internetowych. Zwykle służyła do tworzenia małych lub średnich systemów, ale znane są przypadki, gdy w PHP pisano bardzo duże systemy. Przykładem może być serwis Allegro, który powstawał właśnie w PHP. Z PHP korzystali tacy giganci jak Yahoo!, Wikipedia czy Facebook. Stało się ono też domeną systemów e-commerce (sklepy internetowe, platformy sprzedażowe), zdominowało też blogosferę. W roku 2004 dostępna była już dojrzała, w pełni obiektowa wersja PHP 5.

ASP/ASP.NET

Drugą z dominujących w opisywanym okresie technologii było rozwiązanie oferowane przez giganta informatycznego z Redmond, czyli firmę Microsoft. Active Server Pages (w skrócie ASP) to oparty na języku skryptowym VBScript sposób generowania po stronie serwera dynamicznych stron internetowych. Pierwsza wersja ASP światło dzienne ujrzała w roku 1996, dojrzała wersja ASP 3.0 zaprezentowana została, wraz z serwerem IIS 5.0 w roku 2000. Natomiast w 2002 roku pojawił się jej bezpośredni

następca, bazujący na technologii .NET kompletny, obiektowy framework ASP.NET[15]. Było to kompleksowe, bardzo zaawansowane rozwiązanie łączące w jedną, zwartą całość języki programowania rodziny .NET, serwer WWW, bazę danych Microsoft SQL Server. Wszystko to osadzone w ramach systemu operacyjnego Microsoft Windows Server i wsparte środowiskiem programistycznym Visual Studio.

JAVA/J2EE

W roku 1999 firma Sun Microsystems zaprezentowała kompletną, dojrzałą platformę programistyczną opartą o nowatorski, zaprezentowany kilka lat wcześniej język Java. J2EE (ang. *Java 2 Enterprise Edition*) możemy traktować jako standard tworzenia wielowarstwowych, skalowalnych aplikacji korporacyjnych, czyli gotowych wspierać działalność dużych przedsiębiorstw. Dużą rolę w tym podejściu stanowią komponenty osadzone na serwerze aplikacyjnym. J2EE od początku traktowany był jako standard, zbiór wielu reguł i interfejsów programistycznych, a nie gotowy produkt jednej firmy. Niezależni dostawcy dostarczają swoje produkty (głównie serwery aplikacyjne), które implementują te interfejsy (lub ich podzbiór). Po określonym procesie testowania i certyfikacji mogą oferować je jako produkty zgodne z J2EE. Czyli mamy do czynienia z kompleksowym rozwiązaniem wykraczającym poza granice jednego producenta. Wspomniane interfejsy już od samego początku istnienia tej technologii obejmowały takie obszary tworzenia aplikacji jak: komponenty biznesowe służące do przechowywania danych i realizacji procesów biznesowych działające w środowisku rozproszonym (EJB), dynamiczne generowanie stron internetowych (Servlet, JSP), obsługę baz danych (JDBC), obsługę transakcji (JTA), zdalne wywoływanie metod (RMI), usługi katalogowe (JNDI), obsługę poczty elektronicznej (JavaMail) i wiele innych [4].

W opisywanym okresie na rynku funkcjonowała już kolejna wersja standardu J2EE (1.4, dostępna od listopada 2003). Dostępne były też zgodne ze standardem serwery aplikacyjne, w tym dojrzały i niezawodny, a przy tym darmowy (wówczas) serwer JBoss w wersji 3[16].

4.1.2. Perspektywy technologii, trendy, wsparcie

Przy wyborze samej technologii istotne są też czynniki pozatechniczne. Duże projekty informatyczne są rozwijane i utrzymywane przez kilka, czasem kilkanaście lat. Z tego punktu widzenia liczy się nie tylko wartość danego rozwiązania w momencie rozpoczynania projektu, ale również po upływie wielu lat. Korzystny jest wybór takiego

[15] dotnet.microsoft.com/apps/aspnet (dostęp: 17.08.2021)
[16] jbossas.jboss.org/downloads (dostęp: 17.08.2021)

narzędzia, które będzie rozwijane, utrzymywane i wspierane przez maksymalnie długi czas, nie okaże się „ślepą uliczką" oraz nie zależy od kondycji i pozycji rynkowej jednego producenta. Nie bez znaczenia jest także perspektywa znalezienia personelu, który gotów będzie podjąć pracę w danej technologii w przyszłości. Na rynku informatycznym jest to szczególnie istotne, charakteryzuje się on bowiem wysoką rotacją zatrudnienia.

4.1.3. Wybór technologii głównej

Biorąc pod uwagę wszystkie wymienione czynniki, kierownictwo projektu podjęło decyzję o wyborze ostatniej ze wspomnianych technologii: J2EE.

Oto lista kluczowych argumentów mających na tę decyzję wpływ:

OTWARTOŚĆ

Ponieważ J2EE to standard, twórca aplikacji nie jest uzależniony wyłącznie od jednego producenta/dostawcy. J2EE gwarantuje, że aplikacje mogą być w miarę bezproblemowo uruchamiane na każdym serwerze aplikacyjnym zgodnym z tym standardem, a takich serwerów jest na rynku sporo.

KOSZTY

Już w 2004 roku istniały zaawansowane i stabilne, a do tego darmowe serwery aplikacyjne zgodne z J2EE. Tej zalety nie miały rozwiązania oferowane przez Microsoft.

STOPIEŃ ZAAWANSOWANIA TECHNOLOGII

W porównaniu z PHP standard J2EE oferował znacznie więcej możliwości i gotowych rozwiązań dla twórców aplikacji wielowarstwowych i rozproszonych. Większość typowych problemów dało się zaimplementować za pomocą dostępnych i opisanych interfejsów i API (ang. *Application Programming Interface*).

NIEZALEŻNOŚĆ OD SYSTEMU OPERACYJNEGO

W przypadku J2EE łatwo o płynne przejście pomiędzy popularnymi systemami operacyjnymi, co oznacza, że tworząc aplikację nie musimy podejmować ostatecznej decyzji dotyczącej docelowego systemu operacyjnego. Dotyczy to przede wszystkim platformy, na której działa serwer aplikacyjny, ale podobną swobodę mają też programiści.

JĘZYK PROGRAMOWANIA

W momencie podejmowania decyzji język Java, na którym bazuje J2EE, wydawał się najbardziej zaawansowany i najlepiej udokumentowany.

MODA I TRENDY

Java i otaczające ją technologie były (i są nadal) bardzo popularne. Istnieje na rynku pokaźna liczba programistów, którzy je znają lub chcą się ich uczyć. To istotny argument w przypadku perspektywy konieczności rozszerzania zespołu.

4.1.4. Pozostałe decyzje architektoniczne na etapie rozpoczynania projektu

Standard J2EE jest elastyczny i oferuje tak wiele możliwości oraz ma na tyle szeroką formułę, że jego wybór nie jest pojedynczą decyzją architektoniczną. To bardziej wybór kierunku, początek drogi. Ostateczna architektura i technologia wytwarzania wymaga podjęcia jeszcze wielu decyzji mniejszego kalibru, choć nie można umniejszać ich roli i znaczenia.

Klasyczna, modelowa aplikacja J2EE składa się (cały czas posługujemy się perspektywą roku 2004) z trzech warstw. Są to: warstwa prezentacji, logiki biznesowej i danych. Warstwa prezentacji to część odpowiadająca za interfejs użytkownika. W aplikacjach webowych sprowadza się to przede wszystkim do dynamicznego generowania stron HTML. W warstwie logiki biznesowej modelujemy i implementujemy procesy biznesowe realizowane przez aplikację. Warstwa danych to zwykle baza danych (w czasie rozpoczynania projektu najczęściej relacyjna), w której przechowywane są przetwarzane dane.

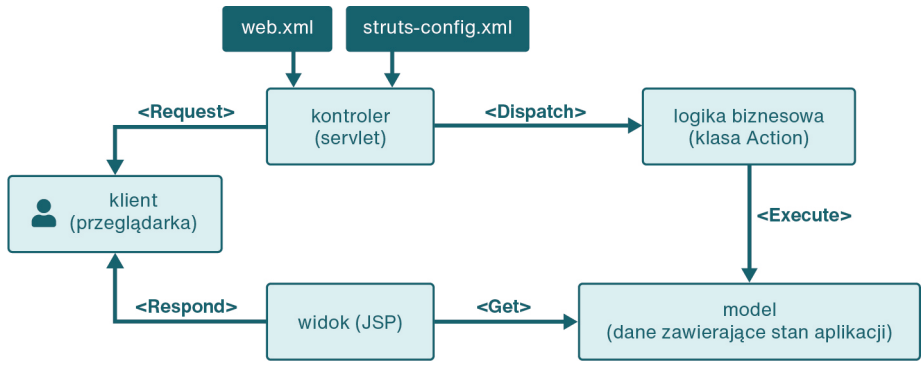
WARSTWY PREZENTACJI I LOGIKI BIZNESOWEJ

W chwili podejmowania decyzji o wyborze sposobu realizacji warstwy interfejsu użytkownika dla systemu OSF jednym z najpopularniejszych, dostępnych i otwartych frameworków dla aplikacji webowych pisanych w języku Java i zgodnych ze standardem J2EE był Apache Struts[17] (wówczas w wersji 1). Biblioteka ta pojawiła się w maju 2000 roku, po 4 latach obecności na rynku była więc produktem sprawdzonym i dojrzałym. Struts można traktować jako framework, który jedynie rozszerza i porządkuje użycie standardowych, istniejących komponentów J2EE. Stanowi propozycję

[17] struts.apache.org (dostęp: 17.08.2021)

usystematyzowanego, dobrze zdefiniowanego i zgodnego z wzorcem MVC (Model-Widok-Kontroler) rozwiązania, które zapewnia jednolity sposób przetwarzania żądań kierowanych przez użytkowników do aplikacji.

Celem niniejszej pracy nie jest szczegółowy opis działania frameworka Apache Struts, dlatego w tym miejscu przypomnimy jedynie w kilku słowach zasadę jego działania: żądanie HTTP wygenerowane przez użytkownika trafia do aplikacji (kontenera servletów) i przekazywane jest na podstawie mapowania do zdefiniowanej akcji (mapowanie znajduje się w specyficznych dla Struts plikach konfiguracyjnych). Akcję możemy traktować jako odpowiednik bądź rozszerzenie servletu. Zadaniem akcji jest wykonanie określonej operacji biznesowej, wygenerowanie modelu (czyli obiektu z danymi) i wyświetlenie tych danych za pomocą wskazanego przez akcję widoku. Do dynamicznego generowania widoku używamy zazwyczaj standardowych stron JSP, choć oczywiście można to zrobić w dowolny, inny sposób. Wyraźnie więc widać, że zadaniem frameworka Apache Struts jest jedynie wprowadzenie pewnego jasno zdefiniowanego standardu, który ustala i formalizuje typowy przepływ danych w aplikacji (por. Ilustracja 4.1).



Ilustracja 4.1. Zasada działania frameworka Apache Struts

Czy w drugiej połowie roku 2004 można było wybrać inne rozwiązanie dla realizacji warstwy interfejsu użytkownika? Można było zrezygnować ze wsparcia Apache Struts, a zadania biblioteki realizować samodzielnie, używając jedynie bazowej technologii JSP (dostępnej od 1999 roku) oraz JSTL (od 2002). Ale to byłaby rezygnacja z możliwości systematycznej obsługi pewnych powtarzalnych, standardowych czynności programistycznych. Inne, bardziej zaawansowane od Apache Struts rozwiązania już istniały, ale ich rynkowy staż był bardzo krótki. Biblioteka JSF (ang. *Java Server Faces*), czyli następca technologii JSP, została zaprezentowana w marcu 2004 roku. Mówimy o specyfikacji w wersji 1.0. Jej użycie wiązałoby się z dużym ryzykiem wzięcia na siebie „chorób wieku dziecięcego", którymi zwykle charakteryzują się nowości.

Podobnym ryzykiem byłoby użycie biblioteki Tapestry[18], która w roku 2004 cały czas była technologią uważaną za młodą. Z perspektywy czasu wybór Apache Struts wydaje się dobry. Obszerne i wyczerpujące porównanie omawianych technologii można znaleźć w pracy Demana Rahmana [26].

Jeśli chodzi o warstwę biznesową, to w pierwszym okresie istnienia projektu zdecydowano się na rozwiązanie, w którym logika biznesowa realizowana jest przez akcje obsługujące żądania, dedykowane klasy biznesowe oraz obiekty typu DAO odpowiadające za współpracę z bazą danych. To rozwiązanie było wystarczające. Nie zdecydowano się na dedykowane obiekty EJB do reprezentowania encji. Czas pokazał, że dość szybko przestały się one cieszyć popularnością, podejście to zyskało opinię zbyt problematycznego i „ciężkiego" (szczególnie w wersjach do 2.1), a ich rozproszony charakter rzadko był rzeczywistą zaletą [13].

WARSTWA DANYCH

Podjęcie decyzji co do wyboru technologii w warstwie danych było stosunkowo proste. Po wstępnej analizie i porównaniu zalet i wad relacyjnych baz danych z innymi, dostępnymi wówczas rozwiązaniami (np. Lotus Notes), po wzięciu pod uwagę potencjalnej ilości danych, liczby rekordów w przyszłości czy możliwego obciążenia systemu wybrano bazę danych Oracle. Wybór relacyjnej bazy wynikał z porównania możliwości technologii. Wybór dostawcy, czyli Oracle, wynikał przede wszystkim z faktu, że Ośrodek Przetwarzania Informacji posiadał już licencję na ten produkt, zatrudniał też osoby, które miały wystarczające doświadczenie w posługiwaniu się tą bazą danych. A kiedy jest się właścicielem najpopularniejszej i najbardziej renomowanej bazy danych na rynku, nie szuka się innej.

4.2. Budowanie zespołu

Kolejnym, bardzo istotnym czynnikiem podczas wyboru sposobu tworzenia systemu informatycznego są technologiczne i społeczne warunki jego funkcjonowania, możliwości stworzenia zespołu posiadającego odpowiednie kompetencje, preferencje oraz doświadczenie. Specjaliści IT odpowiedzialni za tworzenie oprogramowania zaliczani są do nowego segmentu struktury społecznej, tzw. klasy kreatywnej, wyróżniającego się specyficznym stylem i warunkami pracy, potrzebami oraz kompetencjami. Nie bez znaczenia są również możliwości rekrutacji i warunki rynku pracy sektora IT. Temat ten jest na tyle obszerny i istotny, że poświęcimy mu niniejszy podrozdział.

[18] tapestry.apache.org (dostęp: 17.08.2021)

4.2.1. Społeczeństwo informacyjne i klasa kreatywna

Pojęcie „społeczeństwo informacyjne" (SI) powstało na początku lat 60. XX w. w Japonii i w ciągu dekady stało się tam podstawą dla budowy japońskiego przemysłu elektronicznego. Celem planowanych przemian życia gospodarczego i społecznego było wyjście poza wyczerpujący się model imitacji wzorów rozwoju zaczerpniętych ze Stanów Zjednoczonych Ameryki Północnej. W Europie pojęcie to zaczęło funkcjonować w debacie publicznej pod koniec lat 70. XX w., gdy koncepcja społeczeństwa, w którym przetwarzanie informacji jest fundamentem życia społecznego i gospodarczego została sformułowana we Francji [3].

W połowie lat 90. XX w., w raporcie zwanym raportem Bangemanna [28], wskazano rekomendacje dla rozwoju społeczeństwa informacyjnego w Unii Europejskiej. Zadanie budowy społeczeństwa informacyjnego i gospodarki opartej na wiedzy uznane zostało za jeden z głównych obszarów w strategii Lizbońskiej (przyjęta w marcu 2000 roku) [24].

Samo pojęcie „społeczeństwa informacyjnego" bywa krytykowane m.in. za brak zjawiska mogącego być punktem odniesienia w historii rozwoju społeczno-technologicznego [3]. Zdaniem Edwina Bendyka pojęciem, które znacznie lepiej odzwierciedla społeczne skutki daleko idących zmian technologicznych jest zaproponowany przez Manuela Castellsa [7] termin „społeczeństwo sieci". Termin ten odzwierciedla współczesne, jakościowo nowe, globalne relacje będące skutkiem rozwoju technologii. Pojęcia „społeczeństwo informacyjne" i „społeczeństwo sieci" przyjmują czasem wydźwięk utopijnej wizji społeczeństw przyszłości, czy projektu politycznego. Warto podkreślić, że w obu przypadkach zakłada się oparcie nowych relacji na społecznych i ekonomicznych podstawach kapitalizmu. W konsekwencji, obecne podziały i nierówności społeczne organizować by się miały wokół informacji jako centralnego zasobu, nie zaś pieniądza.

Inni autorzy koncentrują się na identyfikacji i prognozowaniu tych nowych podziałów społecznych. Wskazuje się na takie nowe kategorie społeczne jak konsumtariat (konsumpcyjny proletariat), prekariat (ludzie bez perspektyw na poprawę własnej pozycji społecznej i awans społeczny, pracujący w oparciu o niepewne formy zatrudnienia), „klasa twórcza" (związana z przetwarzaniem informacji i symboli), kognitariusze (wysoko wykwalifikowani specjaliści, posiadający ekspercki know how, ale często upośledzeni na rynku pracy), czy netokracja (kontrolująca kluczowe węzły sieci internet) (por. [1]). Z punktu widzenia koncepcji społeczeństwa informacyjnego i społeczeństwa sieci specjalną rolę pełni pojęcie „klasy kreatywnej" [10]. Osoby należące do tej kategorii społecznej charakteryzuje praca polegająca na wytwarzaniu nowych form. W społeczeństwach postindustrialnych miałaby ona stać się podstawą rozwoju gospodarczego. Klasę kreatywną tworzą z jednej strony twórczy profesjonaliści, wykorzystujący zaawansowaną wiedzę do rozwiązywania konkretnych problemów, a z drugiej superkreatywny rdzeń, czyli innowacyjni i kreatywni eksperci, m.in. artyści, naukowcy, badacze, specjaliści IT. Ich główną rolą zawodową jest kreatywność i wytwarzanie innowacji.

Koncepcja kultury kreatywności jako podstawy rozwoju społeczno-gospodarczego spotyka się jednak z krytyką środowisk naukowych, tak na gruncie teoretycznym jak i empirycznym [31]. Niemniej dostrzegalna jest rosnąca rola zawodów twórczych na globalnym i lokalnym rynku pracy. W szczególności dynamiczny rozwój sektora technologii informacyjnych i komunikacyjnych, a zwłaszcza usług i technologii informatycznych.

4.2.2. Zjawiska na rynku pracy w sektorze technologii informacyjnych i komunikacyjnych (ICT)

Przełom XX i XXI w. można określić jako okres wyjątkowej prosperity na polskim rynku pracy sektora IT[19]. Zjawisko to związane było z wysokim popytem na specjalistów uczestniczących w procesach wytwarzania i utrzymania oprogramowania przy niewystarczającej podaży [14]. W warunkach rosnącego zatrudnienia i dynamiki płac, pracownicy sektora IT mogli w znacznym stopniu kreować warunki swojego zatrudnienia. Typowa dla informatyków, wytwórców oprogramowania, kultura pracy w systemie projektowym przyczyniła się do popularności wśród nich pozaetatowych form zatrudnienia.

Według raportu PARP [27, s. 19] na początku drugiej dekady XXI w. w Polsce zatrudnienie w sektorze ICT wynosiło ok. 430 tysięcy osób. Jednocześnie udział tej branży w PKB wynosił około 8%. W latach 2011-2014 liczba firm sektora ICT wzrosła o 24,5%, a liczba pracowników tego sektora rosła o około 6% rocznie. Jednocześnie pod koniec drugiej dekady XXI w. Polska wciąż była atrakcyjnym krajem dla outsourcingu usług informatycznych, gdyż koszty pracy w sektorze ICT były o 45-70% niższe w porównaniu do krajów Europy Zachodniej [27, s. 19].

Według danych GUS liczba przedsiębiorstw z sektora ICT wynosiła w 2016 roku 2278 i wzrosła w 2019 do 2393. W tym okresie najszybciej rosła liczba przedsiębiorstw z obszaru usług ICT, w szczególności usług IT [33, s. 29].

Sytuacji tej sprzyjały procesy postępującej informatyzacji administracji publicznej oraz wykorzystania technologii informatycznych w przedsiębiorstwach.

Pod koniec drugiej dekady XXI w. niemal wszystkie jednostki administracji publicznej (99,8% w roku 2019) wykorzystywały technologię szerokopasmowego dostępu do internetu [33, s. 23]. 85% jednostek administracji posiadało strony internetowe dostosowane do wymogów ustawy o dostępności cyfrowej. Technologie ICT były wykorzystywane w 2019 roku przez administrację publiczną m.in. jako mechanizm

[19] Skrót ICT, z ang. *information and communication technologies*. Na potrzeby tego opracowania przyjmujemy, że pojęcie to dotyczy rodziny technologii dotyczących przetwarzania, gromadzenia i przesyłania informacji w formie elektronicznej. Pojęciem o węższym zakresie znaczeniowym są technologie informatyczne, które obejmują technologie bezpośrednio związane z komputerami i oprogramowaniem.

zwiększania partycypacji obywateli w procesach decyzyjnych poprzez głosowania i konsultacje on-line (20,7% jednostek administracji), narzędzie zarządzania dokumentami (76,7%) i udostępnianie usług (98,6%) oraz danych przestrzennych obywatelom (75,3%) [33, s. 23].

Podobne zjawiska obserwujemy w sektorze przedsiębiorstw. W 2020 roku 98% podmiotów gospodarczych posiadało szerokopasmowy dostęp do internetu [33, s. 24]. Znaczna część (78,3%) udostępniała swoim pracownikom urządzenia przenośne z mobilnym dostępem do internetu. Około dwie trzecie firm wykorzystywało strony WWW do prezentacji swoich produktów, usług i cenników (66,8%). Wśród dużych przedsiębiorstw ok. 40% wykorzystywało technologie internetu rzeczy (IoT). W 2020 roku nieco ponad 25% firm zatrudniało specjalistów z dziedziny ICT, przy czym według danych GUS najbardziej aktywne w tym zakresie były podmioty duże [33, s. 24].

W latach 2016-2019 wyraźnie zarysował się również trend w postaci rosnącej roli usług informatycznych w przychodach firm z sektora ICT. Udział przychodów netto z tego segmentu w przychodach ze sprzedaży przedsiębiorstw z sektora ICT wzrósł z ok. 25% w 2016 do poziomu 34,2% w 2019 roku [33, s. 31].

Wykorzystanie technologii informacyjnych jest obecnie powszechne również w gospodarstwach domowych. Według danych GUS 90% gospodarstw domowych posiadało w 2020 roku dostęp do internetu szerokopasmowego, a wśród osób w wieku 16-74 lat ponad 81% regularnie korzystało z internetu. Udział regularnych użytkowników internetu był najwyższy wśród osób w wieku do 24 lat (99,2%), uczniów i studentów (99,8%) oraz osób z wykształceniem wyższym (98,2%) [33, s. 24].

Wobec rosnącej roli nowoczesnych technologii teleinformatycznych w gospodarce sektor ICT borykał się deficytem pracowników. Raport Ministerstwa Rodziny, Pracy i Polityki Społecznej pt. „Zawody deficytowe i nadwyżkowe w 2015 roku" [19] wskazuje, że na poziomie ogólnokrajowym do zawodów deficytowych należeli tacy specjaliści ICT jak, projektanci aplikacji sieciowych i multimediów, programiści aplikacji, analitycy systemów komputerowych, specjaliści od rozwoju systemów komputerowych. W 2019 roku katalog zawodów deficytowych związanych z sektorem ICT zawierał już dziewięć grup zawodowych. Byli wśród nich specjaliści do spraw sieci komputerowych, programiści aplikacji, projektanci aplikacji sieciowych i multimediów, analitycy systemów komputerowych, administratorzy systemów komputerowych, specjaliści do spraw baz danych i sieci komputerowych gdzie indziej niesklasyfikowani, projektanci i administratorzy baz danych, technicy sieci internetowych, kierownicy do spraw technologii informatycznych i telekomunikacyjnych [19, 20].

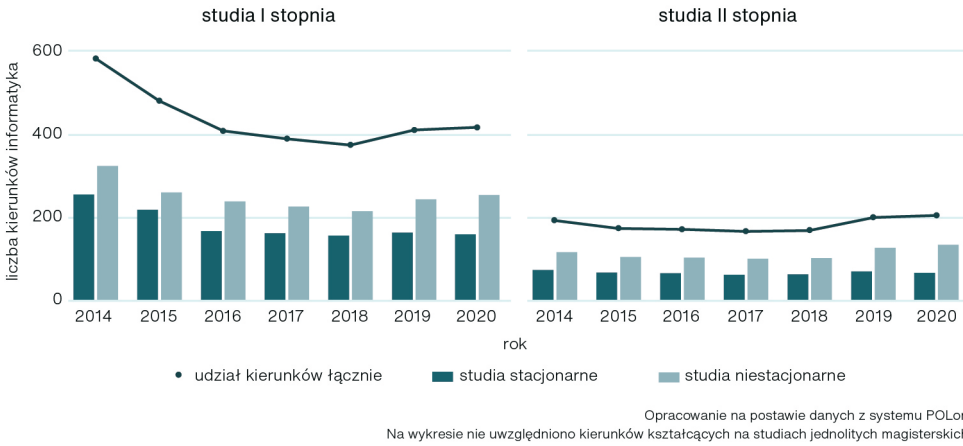
Nieco inny obraz kształtuje się w świetle danych pochodzących z jakościowego badania Barometr zawodów [2]. W oparciu o opinie ekspertów analizujących rynek pracy i posiłkujących się danymi ilościowymi, w każdym z powiatów powstaje jakościowy obraz prognozowanych zawodów deficytowych, zrównoważonych i nadwyżkowych. Wyniki dotyczące powiatów są następnie agregowane na poziomie województw i kraju. Wśród badanych grup zawodów do sektora ICT należą analitycy, testerzy i operatorzy systemów teleinformatycznych oraz projektanci i administratorzy baz danych, programiści.

Obie te grupy w prognozach na kolejne lata z okresu 2016-2020 zaklasyfikowane zostały na poziomie krajowym jako zawody zrównoważone, czyli takie dla których oczekuje się zbliżonej liczby ofert pracy i osób zdolnych i chętnych do podjęcia zatrudnienia w danym zawodzie [2] . Analitycy, testerzy i operatorzy systemów teleinformatycznych należeli do zawodów deficytowych (zawody, w których oczekuje się w perspektywie roku dużego zapotrzebowania na pracowników przy jednoczesnej małej podaży wykwalifikowanych pracowników chętnych do podjęcia zatrudnienia) w województwach: lubuskim (lata 2016 i 2019), podlaskim (w roku 2016), pomorskim (w roku 2016) i wielkopolskim (w roku 2019). Natomiast projektanci i administratorzy baz danych, programiści należeli do zawodów deficytowych w województwach: kujawsko-pomorskim (lata 2018 i 2019), lubuskim (w roku 2016), małopolskim (w latach 2017-2020), podlaskim (w roku 2018), pomorskim (w roku 2016), śląskim (w latach 2017-2018) i wielkopolskim (w latach 2018-2020).

Rynek pracy sektora IT w Polsce boryka się z szeregiem trudności. W opracowaniach branży HR wskazuje się od kilku lat m.in. na dużą fluktuację kadr [25]. Pracownicy poszukują nowych miejsc pracy oferujących lepsze warunki pracy, możliwości awansu i wyższe zarobki. Taki profil pracowników zdaje się być typowy dla przedstawicieli klasy kreatywnej, a w szczególności superkreatywnego rdzenia. Specjaliści IT preferują pracę zdalną, w trybie projektowym, która pozwala na większą elastyczność w gospodarowaniu budżetem czasu. Jednocześnie wskazuje się na wyższy poziom krótkotrwałej absencji w pracy wśród tej grupy pracowników w porównaniu do pozostałych grup zawodowych. Specyfika pracy specjalistów IT polega również na tym, że w przedsiębiorstwach z tej branży zespoły pracowników są zwykle mniejsze niż w ogóle przedsiębiorstw. Wg badania Sedlak & Sedlak [25] na jednego kierownika w sektorze IT przypada o ok. 40% pracowników mniej niż w ogóle badanych firm. To samo badanie wykazało, że w porównaniu do danych na poziomie kraju, przedsiębiorstwa z sektora IT cechowały się niższą rentownością i produktywnością. Jednocześnie zarobki pracowników sektora są znacznie wyższe (por. wyniki badania ekonomicznych losów absolwentów, część 4.2.4).

4.2.3. Kształcenie informatyków i sytuacja absolwentów informatyki na rynku pracy

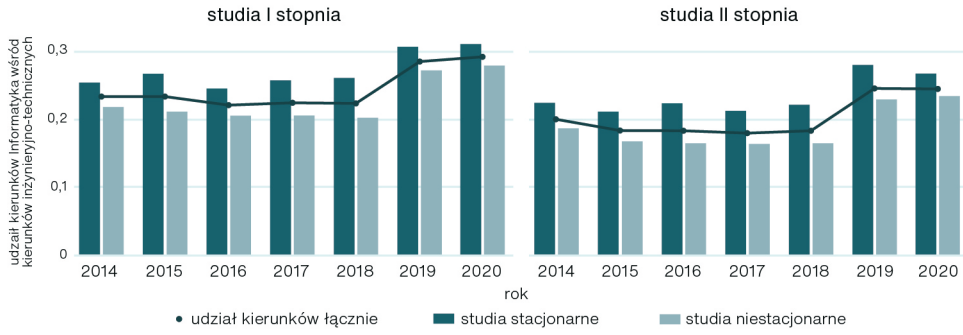
Sytuacji na rynku pracy towarzyszyły zmiany w obszarze szkolnictwa wyższego. Z jednej strony w okresie 2014-2020 zaobserwować można systematyczny spadek liczby kierunków kształcących na specjalistów IT [23]. Oferta dydaktyczna uczelni była bardziej rozbudowana w zakresie studiów I stopnia. Według danych z systemu POL-on w roku 2014 uruchomiono 581 kierunków Informatyka na studiach I stopnia i 194 kierunki na studiach II stopnia (liczba kierunków oznacza liczbę uruchomień kierunków, na których 31 grudnia danego roku studiował przynajmniej jeden student). Liczba kierunków na studiach I stopnia zmniejszała się i w roku 2018 wynosiła 375. Podobny trend, choć o mniejszym nasileniu dotyczył studiów II stopnia (por. Ilustracja 4.2).



Opracowanie na postawie danych z systemu POLon.
Na wykresie nie uwzględniono kierunków kształcących na studiach jednolitych magisterskich.

Ilustracja 4.2. Liczba kierunków Informatyka według poziomu i formy studiów

Jednocześnie, udział kierunków Informatyka wśród kierunków technicznych na studiach I stopnia w latach 2014-2020 wynosił ok. 2-3%, a na studiach II stopnia ok. 2-2,5% (por. Ilustracja 4.3). Ze względu na zmiany ustawowe, dla lat 2014-2018 uwzględniono w obliczeniach dziedzinę nauk inżynieryjno-technicznych, a dla lat 2019-2020 obszar nauk technicznych i obszar nauk ścisłych. Oznacza to, że zmiany liczby kierunków Informatyka w analizowanym okresie odzwierciedlały trend w całym sektorze szkolnictwa wyższego. Udział kierunków Informatyka w ogóle kierunków w latach 2014-2020 na studiach I stopnia wynosił ok. 6%, na studia II stopnia ok. 5%.

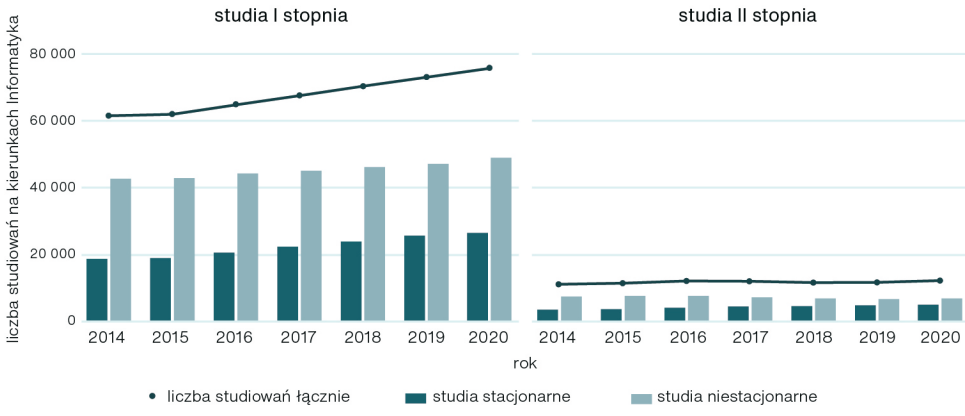


Opracowanie na postawie danych z systemu POLon.
Na wykresie nie uwzględniono kierunków kształcących na studiach jednolitych magisterskich.

Ilustracja 4.3. Udział kierunków Informatyka wśród kierunków inżynieryjno-technicznych w kolejnych latach według poziomu i formy studiów

Spadająca liczba kierunków kształcących specjalistów IT nie oznaczała zmniejszenia się liczby studiowań. Wyraźny wzrost dotyczył studiów I stopnia. W roku 2014 na studiach I stopnia aktywnych było ok. 61,5 tys. studiowań, a w roku 2020 ponad 75,5 tys. (liczba studiowań w danym roku, również oznacza liczbę studiowań na 31 grudnia da-

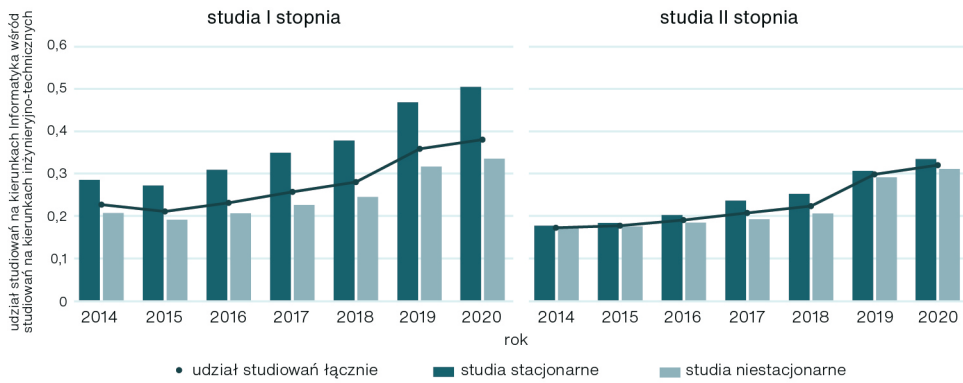
nego roku) (por. Ilustracja 4.4). Jednocześnie w całym sektorze szkolnictwa wyższego od roku 2012 obserwować można trend przeciwny, tj. systematycznie malejącą liczbę studiowań.



Opracowanie na postawie danych z systemu POLon.
Na wykresie nie uwzględniono kierunków kształcących na studiach jednolitych magisterskich.

Ilustracja 4.4. Liczba studiowań na kierunkach Informatyka w kolejnych latach według poziomu i formy studiów

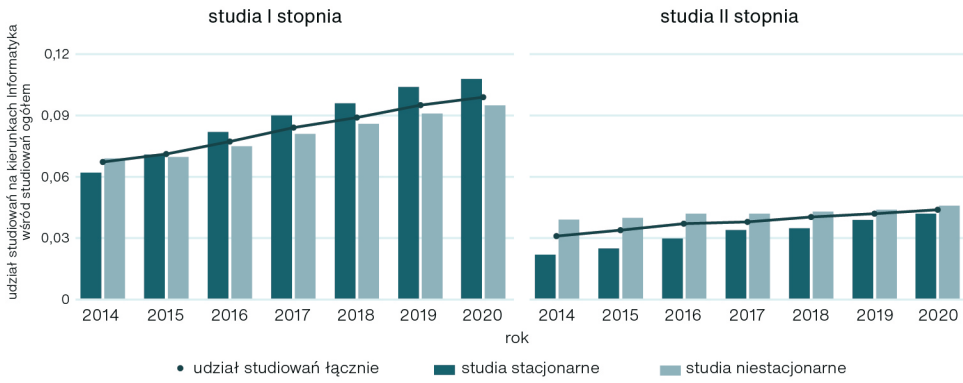
W rezultacie udział studiowań na kierunkach Informatyka wśród kierunków technicznych systematycznie rósł w latach 2014-2020. W roku 2014 wynosił on nieco ponad 20%, a w roku 2020 niemal 40% na studiach I stopnia oraz niecałe 20% w roku 2014 i ponad 30% w roku 2020 na studiach II stopnia (por. Ilustracja 4.5).



Opracowanie na postawie danych z systemu POLon.
Na wykresie nie uwzględniono kierunków kształcących na studiach jednolitych magisterskich.

Ilustracja 4.5. Udział studiowań na kierunkach Informatyka wśród studiowań na kierunkach inżynieryjno-technicznych według poziomu i formy kształcenia

Analogiczny trend dotyczył udziału studiowań na kierunkach Informatyka w ogólnej liczbie studiowań. W roku 2014 na studiach I stopnia wynosił on niecałe 7%, a w roku 2020 niemal 10%. Na studiach II stopnia w roku 2014 udział ten wynosił ok. 3%, podczas gdy w roku 2020 ponad 4% (por. Ilustracja 4.6).

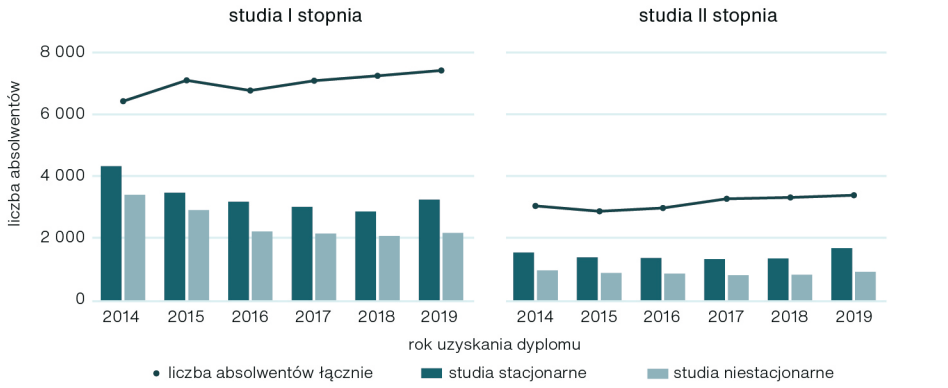


Opracowanie na postawie danych z systemu POLon.
Na wykresie nie uwzględniono kierunków kształcących na studiach jednolitych magisterskich.

Ilustracja 4.6. Udział studiowań na kierunkach Informatyka w ogóle studiowań według poziomu i formy kształcenia

Zjawiska te miały swoje odzwierciedlenie w zwiększającej się podaży pracowników w sektorze IT. Co roku na rynek pracy wchodziły kolejne roczniki absolwentów, przy czym liczba absolwentów kierunków Informatyka systematycznie rosła w latach 2014-2020. Na podstawie danych z Ogólnopolskiego Systemu Monitorowania Ekonomicznych Losów Absolwentów Szkół Wyższych[20] [22] można określić dynamikę tych procesów. W roku 2014 studia I stopnia na kierunkach Informatyka ukończyło z dyplomem 6410 osób, a studia II stopnia 3003 osoby. Pod koniec analizowanego okresu, czyli w roku 2020, liczba absolwentów studiów I stopnia kierunków Informatyka była o ponad tysiąc osób wyższa i wynosiła 7425 osoby. Liczba absolwentów studiów II stopnia w tej grupie kierunków wzrosła o 375 osób (por. Ilustracja 4.7).

[20] ela.nauka.gov.pl/pl (dostęp: 17.08.2021)

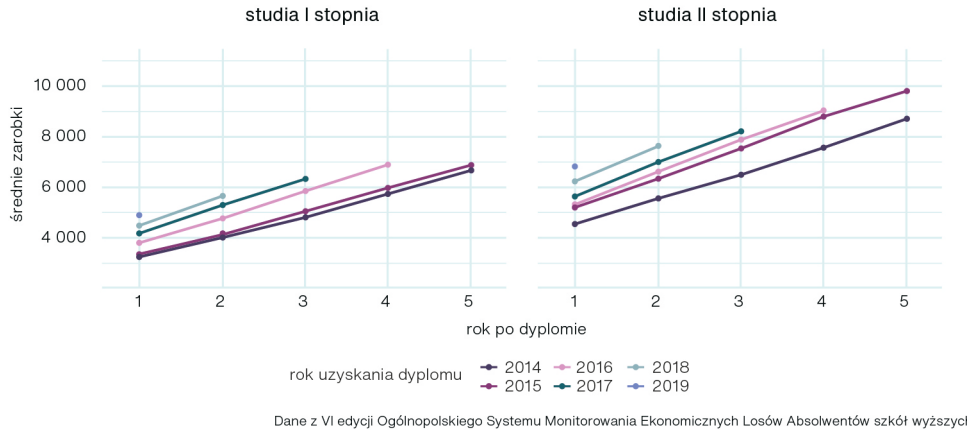


Dane z VI edycji Ogólnopolskiego Systemu Monitorowania Ekonomicznych Losów Absolwentów szkół wyższych.

Ilustracja 4.7. Liczba absolwentów kierunków Informatyka w kolejnych latach według poziomu i formy studiów

4.2.4. Sytuacja absolwentów kierunków Informatyka na rynku pracy

Ogólnopolski System Monitorowania Ekonomicznych Losów Absolwentów Szkół Wyższych (ELA) pozwala śledzić zmiany zarobków absolwentów informatyki w kolejnych pięciu latach po uzyskaniu dyplomu począwszy od absolwentów, którzy uzyskali dyplom w 2014 roku. Zarobki kolejnych roczników absolwentów tych kierunków systematycznie rosły (por. Ilustracja 4.8). Średnie zarobki netto w pierwszym roku po uzyskaniu dyplomu absolwentów studiów I stopnia z rocznika 2014 wynosiły ok. 3200 zł netto, podczas gdy średnie zarobki absolwentów z rocznika 2019 niemal 5000 zł netto. Osoby, które uzyskały tytuł magistra na kierunkach Informatyka w 2014 roku zarabiały średnio ok. 4551 zł netto, a już pięć lat później, w pierwszym roku po uzyskaniu dyplomu, absolwenci studiów II stopnia na kierunkach Informatyka przeciętnie uzyskiwali zarobki netto na poziomie 6805 zł. Poziom wynagrodzeń kolejnych kohort absolwentów analizowanych kierunków rósł w czasie. W piątym roku po uzyskaniu dyplomu absolwenci studiów I stopnia przeciętnie uzyskiwali zarobki na poziomie 6665 zł (absolwenci z rocznika 2014), a więc ponad dwukrotnie więcej niż w momencie, w którym osoby te wchodziły na rynek pracy. Absolwenci studiów II stopnia, którzy uzyskali dyplom w roku 2014 po pięciu latach na rynku pracy średnio zarabiali ok. 8696 zł netto, natomiast absolwenci rocznika 2015 w piątym roku po uzyskaniu dyplomu przeciętnie uzyskiwali zarobki na poziomie 9787 zł netto.

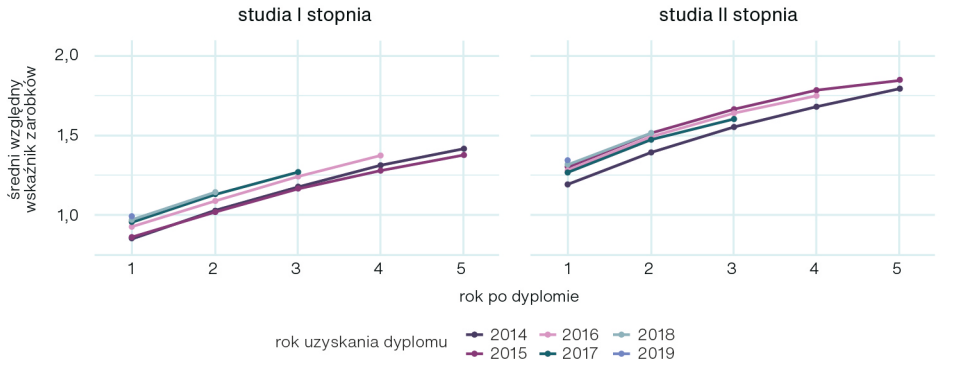


Dane z VI edycji Ogólnopolskiego Systemu Monitorowania Ekonomicznych Losów Absolwentów szkół wyższych.

Ilustracja 4.8. Średnie zarobki ogółem absolwentów kierunków Informatyka w kolejnych latach po dyplomie według poziomu studiów

Informacji o względnej sytuacji na rynku pracy absolwentów kierunków Informatyka dostarcza stosowany w badaniu ELA względny wskaźnik zarobków (WWZ). Miara ta wyznaczana jest jako stosunek średnich zarobków danego absolwenta do średnich zarobków uzyskiwanych w jego powiecie zamieszkania w okresie objętym badaniem. Wartości WWZ poniżej 1 oznaczają, że średnio absolwenci danego kierunku zarabiają poniżej średniej wynagrodzeń w ich powiatach zamieszkania, natomiast wartości powyżej 1 oznaczają, że przeciętnie absolwenci danego kierunku uzyskują zarobki wyższe niż średnie zarobki w ich powiecie zamieszkania.

Dla absolwentów studiów I stopnia kierunków Informatyka w pierwszym roku po dyplomie wartość wskaźnika WWZ w 2014 i 2015 roku wynosiła ok. 0,9, a więc przeciętnie zarabiali oni nieco mniej niż średnie zarobki w ich miejscach zamieszkania (por. Ilustracja 4.9). Jednak w piątym roku po uzyskaniu dyplomu średni WWZ absolwentów z tych roczników wynosi już 1,4, co oznacza znaczny wzrost względnego poziomu uzyskiwanych zarobków. Średnie zarobki absolwentów informatyki na studiach II stopnia były wyższe od przeciętnych wynagrodzeń w powiatach ich zamieszkania już w pierwszym roku po uzyskaniu dyplomu. Dotyczy to wszystkich kohort absolwentów objętych badaniem ELA. W piątym roku po dyplomie średnia wartość WWZ osób z tytułem magistra uzyskanym na kierunkach Informatyka wynosiła już 1,8, a więc ich zarobki były niemal dwukrotnie wyższe niż przeciętne zarobki w ich miejscach zamieszkania.

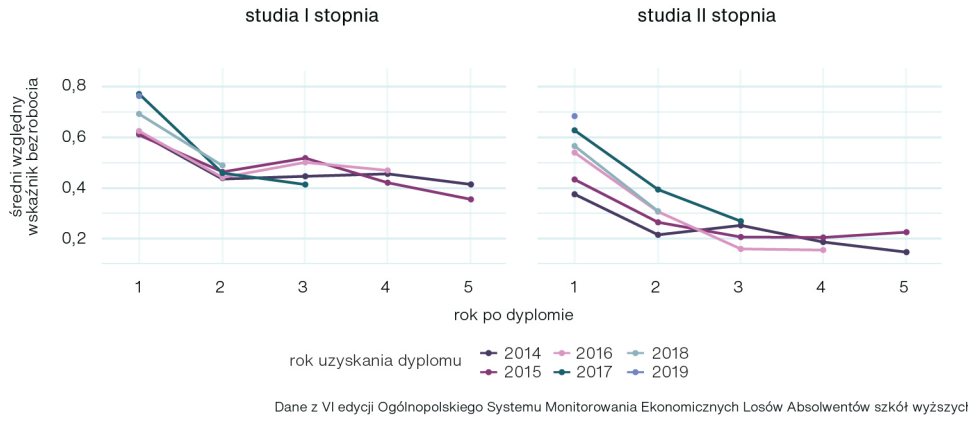


Dane z VI edycji Ogólnopolskiego Systemu Monitorowania Ekonomicznych Losów Absolwentów szkół wyższych.

Ilustracja 4.9. Względny wskaźnik zarobków absolwentów kierunków Informatyka w kolejnych latach po dyplomie

Obraz sytuacji na rynku pracy absolwentów kierunków Informatyka dopełnia informacja o ich doświadczeniu bezrobocia. W badaniu ELA stosuje się w tym celu m.in. względną miarę, czyli stosunek ryzyka bezrobocia do średniej stopy rejestrowanego bezrobocia w powiatach zamieszkania absolwentów w okresie objętym badaniem. Miara ta nosi nazwę względnego wskaźnika bezrobocia (WWB). Wartości poniżej 1 oznaczają niższe ryzyko bezrobocia absolwentów w stosunku do stopy bezrobocia w ich miejscach zamieszkania, a wartości powyżej 1 oznaczają wyższe ryzyko bezrobocia.

Absolwenci studiów I i II stopnia kierunków Informatyka już w pierwszym roku po dyplomie uzyskiwali średnie wartości WWB poniżej 1, a więc ich ryzyko bezrobocia było niższe niż stopa bezrobocia w ich miejscach zamieszkania. Przy tym, dla kolejnych roczników absolwentów wartość wskaźnika WWB w pierwszym roku po dyplomie systematycznie rosła. Średnia wartość WWB dla absolwentów studiów I stopnia z lat 2014 i 2015 wynosiła ok. 0,6, a dla absolwentów studiów II stopnia z tych kohort ok. 0,4. Absolwenci z rocznika 2019 studiów I stopnia średnio uzyskali wartość WWB na poziomie ok. 0,8, a absolwenci studiów II stopnia z tego samego rocznika wartość 0,7 (por. Ilustracja 4.10). Dane te, w kontekście przywoływanych wcześniej prognoz z badania Barometr zawodów, sugerować mogą stopniowe wysycanie się popytu na rynku pracy w sektorze IT. Niezależnie od tych ustaleń należy stwierdzić, że średnio rzecz biorąc sytuacja na rynku pracy absolwentów kierunków Informatyka jest znacznie lepsza niż innych osób aktywnych zawodowo w ich miejscach zamieszkania. W kolejnych latach po uzyskaniu dyplomu absolwenci tych kierunków uzyskują średnie wartości WWB w zakresie 0,5-0,4, w przypadku absolwentów studiów I stopnia lub 0,4-0,1, w przypadku absolwentów studiów II stopnia.



Dane z VI edycji Ogólnopolskiego Systemu Monitorowania Ekonomicznych Losów Absolwentów szkół wyższych.

Ilustracja 4.10. Względny wskaźnik bezrobocia absolwentów kierunków Informatyka w kolejnych latach po dyplomie

4.2.5. Perspektywy

Gwałtowny rozwój cyfrowych technologii informacyjnych i komunikacyjnych w XX i XXI w. zwrotnie oddziaływał z jednej strony na polityki państw poszukujących sposobów na zbudowanie przewagi konkurencyjnej, z drugiej strony na społeczne mechanizmy budowania relacji, zróżnicowania i nierówności społecznych. Nowe podziały społeczne przebiegają według cyfrowych kompetencji do tworzenia i przetwarzania informacji. Zjawiska te znajdują swoje odzwierciedlenie na polskim runku pracy. Obserwuje się rosnącą liczbę przedsiębiorstw w sektorze ICT, których udział w PKB systematycznie rośnie. Rozwój tego sektora wydaje się być odpowiedzią na potrzeby informatyzacji i automatyzacji procesów w przedsiębiorstwach i administracji publicznej, jak i wykorzystanie technologii informacyjnych w gospodarstwach domowych oraz przez użytkowników indywidualnych. Polska jest też atrakcyjnym miejscem dla outsourcingu usług informatycznych. Procesy te skutkują ciągle dużym popytem na wysoko wykwalifikowanych specjalistów IT, pomimo zwiększającej się podaży w postaci rosnącej liczby miejsc na kierunkach kształcących informatyków. Zarobki absolwentów kierunków Informatyka już w kilka lat po uzyskaniu dyplomu znacznie przewyższają przeciętne zarobki w ich miejscach zamieszkania, a ryzyko bezrobocia jest niższe niż przeciętne ryzyko w ich miejscach zamieszkania. Zjawiskom tym towarzyszą problemy rynku pracy w sektorze IT, takie jak duża fluktuacja kadr, wysoki poziom krótkotrwałej absencji w pracy, czy w końcu niższa rentowność i produktywność w porównaniu do ogółu przedsiębiorstw w Polsce. Procesy te określają warunki brzegowe dla decyzji dotyczących tworzenia systemu informatycznego.

4.3. Rozwój i ewolucja systemu

W niniejszym podrozdziale koncentrujemy się na zmianach architektonicznych i technologicznych, rozwoju i ewolucji systemu informatycznego OSF. Chcemy pokazać, jak od strony technicznej zmieniał się system, który jest w ciągłym użyciu od 15 lat. W jaki sposób odpowiadano na potrzeby modernizacji, spełniano nowe wymagania klientów, dostosowywano się do zmieniających się standardów i trendów, dbając jednocześnie o stabilność i niezawodność systemu.

Należy wyraźnie podkreślić, że podrozdział ten ma charakter techniczny, nie opisujemy w nim rozwoju funkcjonalności systemu, zwiększania typu i liczby użytkowników czy oferowanych im przez system możliwości. Skupiamy się na architekturze, technologii oraz, w ograniczonym stopniu, na organizacji i procesie wytwórczym.

4.3.1. Osiąganie równowagi pomiędzy koniecznością modernizacji a stabilnością systemu

Informatyka jest jedną z najszybciej rozwijających się dziedzin nauki i techniki. Tak szybki rozwój, oprócz niezaprzeczalnych zalet, ma także swoje wady. Wyraźnie widać je z perspektywy systemów i aplikacji, które charakteryzują się średnim lub długim czasem życia i muszą być stale dostępne dla użytkowników. Z jednej strony istnieje pokusa, a często wręcz potrzeba, by stosować nowe rozwiązania i technologie, z drugiej strony działania takie stanowią duże ryzyko dla stabilności systemu traktowanego jako całość. Oczywiście w myśl zasady, że „stabilne systemy działają zawsze w oparciu o przestarzałe technologie" można przyjąć założenie, że aplikacja będzie do końca swego istnienia tworzona w taki sposób, jaki wybrany został na starcie projektu, ale czasami jest to po prostu niemożliwe. Wśród przyczyn wymuszających zmianę narzędzi, technologii a czasem wręcz całej architektury znajdują się na pewno:

- Konieczność zmian wynikająca z cyklu wydawniczego narzędzi i bibliotek, które są używane w projekcie. Często pojawiają się nowe, kolejne ich wersje, a to pociąga za sobą zakończenie cyklu życia i wsparcia dla wersji wcześniejszych. Twórca lub dostawca przestaje je utrzymywać (w tym poprawiać zgłaszane błędy) i tym samym jesteśmy zmuszeni do aktualizacji.
- Konieczność zmian wynikająca z presji użytkowników lub rynku. Systemy informatyczne działają w zmieniającym się otoczeniu. Tym otoczeniem może być np. system operacyjny, którego używa klient, rodzaj przeglądarki czy wersja smartfona. Warunkiem koniecznym jest zapewnienie kompatybilności naszych aplikacji z systemem klienta. Ale często wymagania idą dalej, dotyczą trendów w dziedzinie interfejsu użytkownika (mody dotyczące wyglądu), zmiany oczekiwań i przyzwyczajeń użytkowników, rosnących wymagań. Czyli oprócz kryteriów wyłącznie technicznych trzeba uwzględnić też inne.
- Konieczność zmian wynikająca z presji zespołu. Większość osób zajmujących się tworzeniem oprogramowania śledzi postęp w swojej dziedzinie i chce dotrzymywać mu kroku. Używanie w projekcie przestarzałych rozwiązań niesie ryzyko pojawienia

się wśród pracowników znużenia bądź zniechęcenia. Problemem jest też rekrutacja nowych osób do projektów realizowanych w przestarzałych technologiach.

Wśród twórców systemów informatycznych często słyszy się postulat radykalnej przebudowy czy wręcz napisania działającej już aplikacji od początku. Pojawia się on nieśmiało tuż po etapie zakończenia prac nad pierwszą wersją systemu, później tego typu sugestie potrafią się nasilać. Oczywiście stanowisko takie nie jest pozbawione sensu, tworząc bowiem nowy, niebanalny system, który modeluje pewien wycinek rzeczywistości, prace analityczne i programistyczne są też de facto czasem, gdy twórcy systemu uczą się i poznają nową dziedzinę. Nie mają kompletnej wiedzy na początku projektu, nabywają jej w trakcie prac. A brak tej wiedzy, brak doświadczenia powoduje powstawanie kolejnych elementów systemu w kształcie, który może w mniejszym lub większym stopniu odbiegać od optymalnej ich postaci. Znacznie większą wiedzę zespół posiada pod koniec trwania projektu, a wraz z nią pojawia się wśród niektórych osób dyskomfort i chęć zrobienia wszystkiego od początku, w znacznie lepszy sposób. Czy taki pomysł jest dobry i daje szansę powodzenia? Czy jest uzasadniony ekonomicznie? Poniżej postaramy się wykazać, że w większości przypadków absolutnie nie.

Historia zna przypadki, w których postanowiono stworzyć nowe wersje istniejących, cieszących się powodzeniem produktów nie na bazie tego, co istniało, ale przez stworzenie ich od początku. W większości był to strategiczny błąd. Często też powód upadku całych firm.

Przykładem niech będzie przeglądarka internetowa Netscape 6.0. Jej publiczna wersja beta pojawiła się trzy lata po wypuszczeniu na rynek wersji 4.0. Wersji 5.0 w ogóle nie było. Trzy lata na rynku przeglądarek internetowych to cała wieczność. Czemu stworzenie nowej wersji trwało tak długo? Ponieważ firma postanowiła napisać kod na nowo. To był praktycznie koniec firmy Netscape [32].

Jedna z najbardziej zasłużonych i jednocześnie innowacyjnych firm software'owych, Borland, swego czasu lider narzędzi programistycznych, postanowił zaistnieć na rynku pakietów biurowych. A gdy popularność zaczął zdobywać system Windows, zarząd postanowił napisać dla niego, zupełnie od nowa wersję bazy danych dBase i arkusza kalkulacyjnego Quattro Pro. Gdy produkty te pojawiły się w końcu na rynku, ten zdominowany był już przez produkty Microsoft. To był też początek końca firmy Borland.

Oczywiście nikt nie kwestionuje konieczności rozwoju i zmian, ale zbyt pochopne krytykowanie istniejących rozwiązań czy technologii wyłącznie dlatego, że istnieją nowsze, nie jest uzasadnione. Duże, działające systemy nie mogą być zmieniane zbyt gwałtownie. Jedyną rozsądną ścieżką ich rozwoju wydaje się być przemyślana ewolucja. Inaczej całe przedsięwzięcie narażamy na wielkie ryzyko o charakterze technicznym, ale też ekonomicznym.

4.4. Odejście od monolitu i nowa architektura

System, którego historia zaczęła się w roku 2005, przez kilkanaście kolejnych lat rósł i ostatecznie osiągnął rozmiary, stopień złożoności pozwalający zaklasyfikować go do grupy systemów wagi ciężkiej. Niestety wady dużej, ciężkiej aplikacji tworzonej i wdrażanej jako całość zaczęły się stawać coraz wyraźniejsze. Wady te wymienimy już za chwilę, podczas opisywania cech aplikacji monolitycznych. Dlatego w okolicach roku 2018 przesłanki i postulaty wprowadzenia istotnych zmian architektonicznych zaczęły nabierać realnych kształtów. Zmiany, o których tutaj mowa, nie stoją w sprzeczności z ewolucyjnym sposobem rozwijania systemu, opisanym w poprzednim podrozdziale. Zmiany architektury muszą odbywać się ewolucyjnie, ponieważ system OSF musi być cały czas dostępny dla bieżących naborów oraz pozwalać na obsługę i rozliczenia projektów, które były zainicjowane wiele lat temu. Nie można zatrzymać systemu na czas modyfikacji. Taki tryb nie kłóci się także ze stosowaną przez kierownictwo projektu bezpieczną strategią rozwoju, unikaniem gwałtownych zwrotów i niesprawdzonych rozwiązań. To podejście przez wiele lat okazywało się skuteczne. Ale projekt w swej ewolucji osiągnął punkt, w którym niezbędna stała się jego dekompozycja. Rozmiar projektu w połączeniu z jego monolityczną architekturą okazał się być źródłem coraz większej liczby problemów i uciążliwości, zalety takiego rozwiązania zaczęły mierzyć się z coraz bardziej widocznymi wadami. Konieczny okazał się podział systemu na części. Nie rewolucja, ale rozsądna dekompozycja na mniejsze elementy, charakteryzujące się znaczną autonomią. Z myślą o programistach, ale również pod kątem niezależnego wdrażania aplikacji.

Oczywiście biorąc pod uwagę panujące na rynku trendy dość naturalnym kierunkiem wydawać się mogła zamiana modelu monolitycznego na mikrousługi. Przynajmniej w teorii wizja takiej migracji wygląda atrakcyjnie.

Zmianom tym i związanym z nimi decyzjom poświęcamy podrozdział niniejszy. Zacznijmy od oceny i porównania architektury monolitycznej i mikrousług.

4.4.1. Architektura monolityczna

Aplikacja stworzona w architekturze monolitycznej charakteryzuje się przede wszystkim autonomią. Jest osobną, pojedynczą jednostką wdrożeniową, zwykle występuje w postaci jednej paczki (archiwum). Wewnątrz tego archiwum znajdziemy wszystkie logiczne komponenty systemu, w przypadku aplikacji trójwarstwowych są to warstwy prezentacji, logiki biznesowej i danych. Wszystkie one są ze sobą ściśle powiązane. Oczywiście aplikacja taka może współpracować z innymi systemami, usługami czy bazami danych, ale rdzeń i wszystkie kluczowe jej komponenty działają w ramach jednego procesu i są wdrażane jako całość [30].

4.4.2. Problemy z monolitem

Najbardziej znane i typowe problemy z architekturą monolityczną możemy podzielić na następujące kategorie:

SKALOWANIE

Osoba postawiona przed zadaniem skalowania aplikacji monolitycznej (np. w celu podniesienia wydajności któregoś z jej elementów) nie ma szerokiego pola manewru i skazana jest na konieczność skalowania całości, czyli wszystkich składników równocześnie i praktycznie w tym samym stopniu (skalowanie jednowymiarowe). Nie istnieje możliwość indywidualnego skalowania wyłącznie najbardziej obciążonych elementów aplikacji, nawet wówczas, gdy jesteśmy w stanie je dokładnie wskazać.

WDRAŻANIE

Aplikacja monolityczna, niezależnie od wewnętrznej struktury i liczby oferowanych funkcji wdrażana jest jako całość. Zwykle jest to jeden plik stanowiący archiwum bądź jeden katalog. Nie jest możliwa niezależna wymiana wybranych modułów czy funkcji. Jeśli choć jeden, nawet niewielki składnik aplikacji działa niepoprawnie, stajemy przed koniecznością wycofania lub ponownego wdrożenia całości.

ZALEŻNOŚĆ OD JEDNEJ TECHNOLOGII

Rozpoczęcie projektu wiąże się z wyborem technologii, w której będzie on realizowany. Czas życia złożonych projektów jest długi i zwykle składa się z co najmniej kilku etapów. W projektach monolitycznych zmiana technologii, w jakiej oprogramowanie jest wytwarzane jest trudna i ryzykowna, dotyczy bowiem całego projektu. Na późniejszych etapach rozwoju wręcz niemożliwa, a to zwykle wówczas pojawia się coraz większa potrzeba wprowadzenia zmian. Im bardziej aplikacja odstaje od obowiązujących w danym momencie standardów i możliwości, które one oferują, tym bardziej odczuwamy potrzebę i potencjalne korzyści z ich zastosowania.

CIĄGŁA INTEGRACJA/CIĄGŁE WDRAŻANIE

Systemy stanowiące monolit można oczywiście rozwijać przy użyciu różnych metod, włączając w to popularne metodyki zwinne. Można na przykład stosować podejście zwane w skrócie CI/CD. Na pewno w teorii.

Praktyka pokazuje nam jednak, że jest to co najmniej mało efektywne i czasem może sprawiać wrażenie działania na siłę. Co w takim razie stoi na przeszkodzie?

System monolityczny sprzyja tworzeniu wielu zbędnych zależności między komponentami aplikacji. Nawet jeśli tego nie chcemy i unikamy, to w wielu przypadkach pokusa pójścia na skróty jest większa niż postanowienie przestrzegania dyscypliny. Szczególnie w warunkach, gdy system rozwijany jest pod presją czasu. Decydujemy się na jeden wyjątek (kompromis), po jakimś czasie na kolejne i tak już zostaje. A liczba zależności w konstrukcji systemu przekłada się na liczbę zależności związanych z udostępnianiem kolejnych funkcji systemu. Nie da się często zaoferować gotowych funkcji, gdy zależą one od innych, niegotowych.

Wielkość aplikacji ma wpływ na czas jej budowania i czas samego wdrażania, złożoność tego procesu, czas samego wdrażania, a co za tym idzie, długość przerw w dostępności systemu. Przypominamy, że ten proces należy przeprowadzić po dowolnej, nawet najmniejszej zmianie.

EFEKTYWNOŚĆ NOWYCH CZŁONKÓW ZESPOŁU

Pojedyncza, rozbudowana baza kodu źródłowego z ogromem wewnętrznych zależności stanowi poważną barierę dla osób, które dołączają do projektu. Nowe osoby pełną efektywność pracy w takich przypadkach osiągają po kilku tygodniach, czasem miesiącach żmudnego i obfitującego w wiele frustracji okresu wdrażania.

4.4.3. Mikrousługi

Nie istnieje jedna, precyzyjna i ogólnie obowiązująca definicja mikrousługi (mikroserwisu, ang. *microservice*). Architektura ta jest wynikiem ewolucji, a samo pojęcie, jak twierdzi Martin Fowler, zaistniało szerzej na warsztatach architektów oprogramowania zorganizowanych niedaleko Wenecji w maju 2011 roku [11]. Przy czym słowem „mikrousługa" opisano architekturę, którą część z uczestników już od pewnego czasu się zajmowała.

Można także przyjąć, że mikrousługi są rozwinięciem architektury zorientowanej na usługi, czyli SOA (ang. *Service Oriented Architecture*), a modyfikacja polega na tym, że szczególny nacisk kładzie się w nich na niewielki rozmiar współpracujących komponentów. W każdym razie próżno szukać standardu mikrousługi, który zostałby zatwierdzony i opublikowany przez określoną instytucję czy grupę osób. To bardziej naturalnie wykształcona koncepcja a nie specyfikacja czy standard.

Na wielką popularność mikrousług wpływ na pewno miały duże firmy (takie jak eBay, Amazon czy Netflix), które z powodzeniem przekształciły swoje potężne, monolityczne aplikacje w zbiory mikrousług.

CZYM JEST MIKROUSŁUGA

Mikrousługa (mikroserwis) to mała aplikacja, komponent, który realizuje jedno, niewielkie zadanie. Czyli zgodnie z zasadą pojedynczej odpowiedzialności [17] koncentruje się na poprawnym wykonywaniu jednej funkcji. Mikroserwis jest niewielki, łatwy w rozwijaniu i utrzymaniu, niezależny i autonomiczny. I co bardzo ważne, można go wdrażać niezależnie od innych mikrousług czy systemów. Każdy mikroserwis oferuje swoje API, za pomocą którego dostarcza swoją usługę klientom, czyli komponentom z niego korzystającym.

Ponieważ mikrousługi komunikują się za pomocą standardowych protokołów i publikowanych API, nie istnieją żadne ograniczenia dotyczące tego, w jaki sposób je implementujemy. Możemy więc używać różnych języków programowania i technologii, wybierać je pod kątem ich przydatności dla realizacji konkretnego zadania. Kod mikroserwisu znajduje się w osobnym repozytorium kodu źródłowego, budujemy na jego podstawie niezależny artefakt, który jest osobno wdrażany i testowany. Ryzyko związane z użyciem nowej technologii jest niewielkie, zwykle kładziemy na szali maksymalnie kilka tygodni pracy niewielkiego zespołu. Taka elastyczność pozwala nam lepiej dobierać narzędzia i tworzyć lepsze i wydajniejsze produkty.

Jeśli kompletny system informatyczny porównamy do kadłuba dużego statku, możemy wyodrębnić w nim pionowe przegrody (grodzie), które zamykają przestrzeń kolejnych segmentów konstrukcji. Jest to jeden z podstawowych systemów zabezpieczeń. Gdy w jednym z obszarów dojdzie do awarii, nie rozprzestrzenia się ona na zewnątrz. Granice odpowiedzialności mikroserwisów możemy porównać do takich grodzi. W przypadku wystąpienia problemu jesteśmy w stanie go szybko wyizolować i przystąpić do naprawy, podczas gdy reszta systemu funkcjonuje bez większych zakłóceń.

Dzięki mikroserwisom mamy dowolność skalowania usług (czyli w większości przypadków zwiększania ich wydajności). Możemy wybrać i skalować tylko te składniki, które tego wymagają. W przypadku aplikacji monolitycznej jej elementy nie mogą być skalowane oddzielnie. Co więcej, typowe mikrousługi typu REST mają charakter bezstanowy, co sprzyja prostej i skutecznej metodzie skalowania poziomego [34].

Niewielki rozmiar kodu składającego się na mikrousługę redukuje prawdopodobieństwo wpływu dokonanej w kodzie zmiany na inne części jej samej. Co więcej, ryzyko związane z wdrożeniem i publikacją nowej wersji artefaktu jest również znacznie mniejsze niż w przypadku aplikacji monolitycznej. Rozmiar mikrousług zmniejsza czas potrzebny na wykonanie pełnego zestawu testów regresji. Łatwiej też wycofać feralną zmianę.

Duża aplikacja monolityczna to ogromna liczba wierszy kodu źródłowego, która nie tylko dla początkującego programisty może stanowić wyzwanie. Dużo łatwiej programistom pracuje się nad projektami, które nie są tak obszerne i skomplikowane, mały rozmiar sprzyja przejrzystości i łatwości implementacji, a w rezultacie jakości tworzonego rozwiązania. Mikroserwisy naturalnie odpowiadają pracy w niewielkich, autonomicznych zespołach, które zwykle są bardziej elastyczne i wydajne, niż ich większe odpowiedniki.

WADY MIKROSERWISÓW

Oczami programisty, który po przejściu na architekturę mikroserwisów będzie odpowiadał za jasno zdefiniowany, niewielki i łatwy w utrzymaniu komponent, zmiana taka ma właściwie same zalety. Niestety z perspektywy całego systemu rachunek zysków i strat nie jest już tak jednoznaczny [21].

Zarządzanie pojedynczym mikroserwisem, jego rozwój i utrzymanie jest znacznie prostsze niż utrzymanie monolitycznego systemu. Ale zarządzanie dwiema setkami takich mikrousług, z których każda oferuje określone API w konkretnej wersji i korzysta z kilku API innych serwisów, których jest klientem, nie jest już zadaniem trywialnym. Wręcz przeciwnie, całość wymaga wielu czynności zapewniających synchronizację, a to stanowi wyzwanie.

Testowanie całości jest również bardzo skomplikowane. Szczególnie w przypadku testowania złożonych procesów biznesowych, których realizacja stanowi łańcuch wielu wywołań współpracujących ze sobą mikroserwisów. Znalezienie błędu logicznego w takim łańcuchu, jego powtórzenie, analiza wielu logów lub zapisów do osobnych lub wręcz różnych koncepcyjnie baz danych stanowi wyzwanie.

Wdrożenie pojedynczego mikroserwisu zwykle jest bardzo proste. Ale wdrażanie aplikacji, na którą składa się całe ich stado, automatyzacja tego procesu, jego monitorowanie i optymalizacja również jest operacją o wysokim stopniu złożoności.

Mikrousługi często używają własnych, oddzielnych baz danych. Wymaga to od nas nowego podejścia do pojęcia transakcji biznesowej, tzn. odejścia od typowych transakcji realizowanych przez bazę danych. Zapewnienie spójności danych nie może być oparte na wykonaniu (lub nie) jednej, atomowej operacji. Spójność trzeba zapewnić samodzielnie, na poziomie logiki aplikacji. Dla wielu programistów jest to zmiana ważnego paradygmatu (atomowa transakcja bazodanowa) i wieloletnich przyzwyczajeń.

4.5. Złoty środek, nowa architektura OSF

Architektura mikroserwisów, tak od pewnego czasu modna i popularna, nie jest idealna. Jej użycie ma wiele zalet, ale wiąże się z kosztami i zagrożeniami, sama w sobie nie jest wcale gwarancją sukcesu. Szczególnie w sytuacjach, gdy nie mamy na tym polu wielu doświadczeń.

A jeśli użycie mikroserwisów nie oferuje tylko i wyłącznie korzyści, to może też monolit nie jest architekturą charakteryzującą się samymi wadami? Przecież przez wiele lat systemy były tworzone właśnie w ten sposób. I może najlepszym wyjściem jest rozwiązanie pośrednie, w którym łączymy dobre strony monolitu i mikrousług, dopasowując je do naszych potrzeb, a nie aktualnej mody i uwzględniając realia projektu, możliwości organizacyjne i dostępne zasoby? Tym tropem podążył zespół OSF.

4.5.1. Czy monolit ma wyłącznie wady

Będąc w tym miejscu dokonano pewnego resetu. Postanowiliśmy, jako zespół, spojrzeć na omawiany problem raz jeszcze, bez żadnych uprzedzeń, preferencji i założeń wstępnych [8].

Zaczęliśmy od krótkiego podsumowania stanu, w jakim się znaleźliśmy oraz zdefiniowaliśmy kluczowe cele, które chcieliśmy osiągnąć. Podstawowe wady systemu to fakt, że od samego początku do chwili obecnej stanowi ogromną, pojedynczą aplikację. Wdrażaną jako całość, trudną w utrzymaniu, budowaniu, a dla osób z mniejszym stażem w projekcie, także w zrozumieniu. W kodzie znajduje się mnóstwo zależności i powiązań. Tych niezbędnych oraz tych zupełnie niepotrzebnych, wprowadzających bałagan, nieprzewidziane i niepożądane interakcje oraz trudności w dalszym rozwoju.

A co chcieliśmy osiągnąć? To właśnie był kluczowy moment, który rzutował na dalszą część historii systemu. Młody pasjonat najnowszych technologii i trendów odpowiedziałby bez wahania: „chcemy zmiany architektury z monolitu na mikroserwisy".

Natomiast zespół architektów systemu OSF udzielił odpowiedzi innej: chcemy wyeliminować wymienione wcześniej, podstawowe wady systemu, ale w sposób równie skuteczny jak odpowiedzialny. Zapewniający niezachwianą i stabilną pracę systemu, redukcję ryzyka i płynne przejście dla wszystkich członków zespołu.

Nie chcieliśmy na siłę wpisywać się w określony (modny) trend, chcieliśmy dostępne na rynku techniki i metody zastosować w takim stopniu i w taki sposób, by przyniosły nam wymierną korzyść.

Zastanówmy się raz jeszcze. Istotą mikroserwisów, podstawową ich zaletą jest dekompozycja. Odpowiedni poziom granulacji jest kluczem do sukcesu. Ale stanowi jednocześnie największe wyzwanie. Nie istnieje bowiem żadna metoda, żaden algorytm, który powie nam, jak głębokich podziałów należy dokonać, jak duże (lub małe) mają być składniki aplikacji. To wymaga doświadczenia, pewnego wyczucia, które pojawia się z czasem i liczbą popełnionych wcześniej błędów. Błędy można popełnić na poziomie szkicowania architektury, ale błędy też mogą popełnić (i na pewno popełnią) programiści, którzy będą zmuszeni zmienić sposób patrzenia na proces tworzenia aplikacji. W tę zmianę trzeba zainwestować czas, ponieść koszty, wkalkulować spadek jakości i wydajności w okresie przejściowym. Zmiana jest potężna i ryzykowna.

Z drugiej strony warto zdawać sobie sprawę z tego, że popularne ostatnio twierdzenie, że monolit oznacza brak architektury, jest rażącym nadużyciem. Aplikacja monolityczna może być przysłowiową wielką kulą błota (ang. *big ball of mud*), często nią jest, ale wcale być nią nie musi. Monolit oznacza po prostu wspólne repozytorium i (zwykle) jeden generowany z niego artefakt. Ale system taki może być jednocześnie świetnie skonstruowany z punku widzenia architektury. Aplikacja monolityczna jak najbardziej może być zbiorem luźno ze sobą połączonych składników, znakomicie zaprojektowanych i komunikujących się za pomocą dobrze dobranych interfejsów.

4.5.2. Modularny monolit

W tym momencie dochodzimy do pojęcia, które okazało się kluczem do dalszego rozwoju naszego systemu. Tym pojęciem jest *modularny monolit* [9].

Modularny monolit to pośredni wariant architektoniczny. Łączymy w nim wyraźny podział na moduły, z których każdy realizuje wyraźnie zdefiniowaną funkcję systemu, wyznaczamy wyraźne granice odpowiedzialności oraz zasady współpracy z innymi modułami/podsystemami. Nie czynimy z nich jednak oddzielnych jednostek wdrożeniowych. Aplikacja, która implementuje wewnątrz wiele serwisów wdrażana jest jako całość. W koncepcję tą znakomicie wpisuje się stwierdzenie Martina Fowlera, który twierdzi, że nigdy nie powinno się rozpoczynać projektu od tworzenia mikroserwisów. Zaczynajmy od monolitu, pamiętajmy o jego modularności, wydzielajmy jego części do mikroserwisów dopiero w chwili, gdy monolit jako taki zaczyna być problemem [12].

Oznacza to, że postanowiliśmy dokonać głębokiego refaktoringu aplikacji oraz jej dekompozycji, ale z użyciem drogi pośredniej. Wyraźnie też dokonaliśmy podziału na moduł widziany jako niezależna jednostka oczami architekta oraz jednostka wdrożeniowa. Mówiąc inaczej, wdrażana jako całość jednostka aplikacji wewnętrznie może składać się z wielu modułów. Ważne jest, aby były one poprawnie zaprojektowane, stanowiły oddzielny byt, komunikowały się za pomocą dobrze zdefiniowanych interfejsów i nie pozwalały w korzystaniu z nich do żadnych dróg na skróty. Dzięki temu praktycznie w każdej chwili, jeśli zajdzie taka potrzeba, możemy wydzielić taki moduł do osobnej jednostki, osobnego serwisu wdrażanego niezależnie. Ale jeśli nie ma takiej potrzeby, nie chcemy niczego skalować na innych od reszty zasadach, nie wprowadzamy niepotrzebnego narzutu technicznego i organizacyjnego.

Postanowiliśmy, że aplikacja w nowej postaci będzie składała się z czterech głównych, oddzielnych, osobno wdrażanych części (zamiast używanego dotychczas jednego artefaktu). Tymi częściami będą osobne aplikacje obsługujące trzech podstawowych klientów oraz moduł centralny, czyli aplikacja, która zawiera zbiór wspólnych, niezależnych od specyfiki klienta usług. Moduł centralny najbardziej wpisuje się w definicję modularnego monolitu, co oznacza, że wewnątrz składa się z oddzielnych, niezależnych komponentów, które w razie potrzeby w prosty sposób mogą być z modułu centralnego wydzielone i wdrażane osobno. Nie robimy tego jednak bez wyraźnej potrzeby.

Oczywiście oprócz czterech podstawowych składników na całość systemu składa się więcej elementów, system korzysta bowiem z innych podsystemów pozwalających na prawidłową integrację z otoczeniem zewnętrznym.

4.5.3. Warstwa utrwalania danych

W nowej wersji systemu zdecydowaliśmy się też na istotną zmianę w warstwie obsługi bazy danych. Nadal korzystamy ze standardowej, relacyjnej bazy danych Oracle, ale sposób jej użycia jest zupełnie inny. Dotychczas dokumenty (czyli podstawowe struktu-

ry danych przetwarzane przez system) zapisywane były w bazie w klasyczny, relacyjny sposób, czyli poprzez odwzorowanie pól dokumentów na pola wielu tabel. Ponieważ struktura większości dokumentów jest bardzo złożona, podobnie złożona musiała być struktura przechowującej ją bazy danych. A ponieważ kolejne wersje dokumentów często różnią się strukturą od poprzednich, oznaczało to konieczność częstych rozszerzeń i modyfikacji bazy. Takie rozwiązanie ma też ujemny wpływ na wydajność systemu. Odwołanie do dokumentu z poziomu aplikacji często wiązało się z koniecznością odczytu danych z wielu tabel bazy.

W nowej odsłonie OSF dokument występuje jako struktura powiązanych ze sobą obiektów zdefiniowanych w języku Java, który do bazy zapisywany jest (i odczytywany) jako całość, zapisywana w formacie JSON. Czyli jeden dokument, niezależnie od stopnia złożoności znajduje się w bazie danych w jednym wierszu (i jednym polu) tabeli bazy danych. Operacja jego zapisu/odczytu jest pojedynczą operacją na bazie.

4.5.4. DDD

Kolejnym trendem w tworzeniu aplikacji stało się podejście nazwane w skrócie DDD (ang. *Domain Driven Design*). Jest to termin ogólnie znany i popularny, nie będziemy go w tym miejscu omawiać, wykracza to bowiem poza ramy niniejszej monografii. Warto natomiast podkreślić, że w nowej architekturze projektu OSF stosujemy wiele pojęć i rozwiązań zgodnych z DDD, oczywiście w zakresie, który odpowiada innym założeniom architektonicznym projektu.

Podstawowa organizacja kodu, który w większości odpowiada za obsługę i przetwarzanie dokumentów zgodna jest z większością reguł i zaleceń DDD. Staramy się w pełni odzwierciedlać w implementacji założenia i obsługiwane procesy biznesowe. Dbamy o ujednolicenie pojęć i struktur, którymi posługuje się klient biznesowy, analityk oraz zespół tworzący kod źródłowy.

Na etapie projektowania i implementacji posługujemy się zdefiniowanymi w DDD kontekstami ograniczonymi (ang. *bounded context*). Kontekstem takim jest pewien ograniczony obszar działalności organizacji, w ramach którego obowiązuje tzw. język wszechobecny (ang. *ubiquitous language*). Jest to wykorzystywany przez dany zespół zbiór jednoznacznych (w obrębie tego zespołu) pojęć i terminów charakterystyczny dla określonej dziedziny biznesowej.

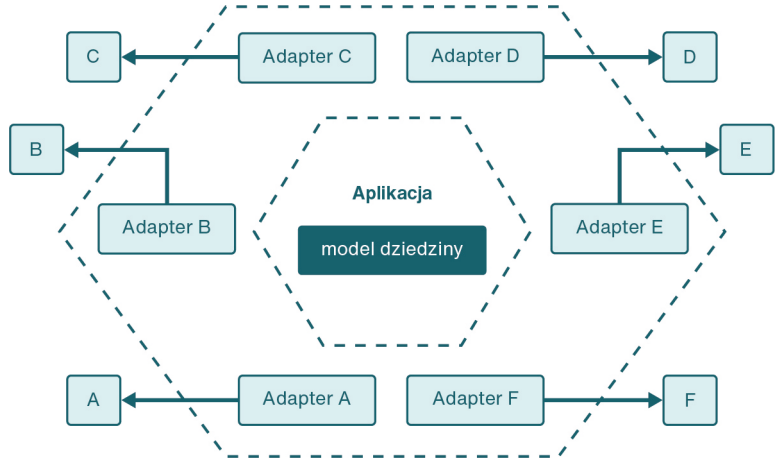
W przypadku systemu OSF kontekstami ograniczonymi są moduły wniosków. Są one od siebie niezależne. Istnieją także konteksty ograniczone niezwiązane bezpośrednio z wnioskami, często mają one bardziej uniwersalny charakter. Wśród nich znajdziemy obsługę użytkowników systemu, bazę jednostek organizacyjnych, mechanizmy administracyjne systemu.

Kontekst ograniczony nie funkcjonuje wyłącznie na poziomie pojęciowym. Znajduje on odzwierciedlenie w strukturze pakietów w kodzie źródłowym, gdzie widoczny jest wyraźny podział na dziedziny, poddziedziny i konteksty ograniczone.

Jedną z największych zalet DDD jest niezależność od konkretnej architektury. Tak więc można łączyć korzyści płynące ze stosowania założeń i technik Domain Driven Design oraz odpowiadającą naszym potrzebom architekturę.

W zespole OSF zdecydowaliśmy się na użycie architektury sześciokątnej. Jest to rozwiązanie będące ewolucją znanej od wielu lat architektury warstwowej, do której dodano elementy zasady odwracania zależności (ang. *Dependency Inversion Principle*, *DIP*). Architektura sześciokątna zwana jest także architekturą portów i adapterów. Więcej szczegółów na ten temat znaleźć można w pracach Vaughna Vernona [39, 40].

Organizacja typowego kontekstu ograniczonego nowej architektury systemu OSF odpowiada założeniom architektury sześciokątnej (por. Ilustracja 4.11).



Ilustracja 4.11. Architektura sześciokątna

W kodzie występuje podział na następujące warstwy (pakiety):

domain – z punktu widzenia DDD centralne miejsce aplikacji, które zawiera definicje klas obiektów domenowych. Klasy te tworzą agregaty, które zawierają podstawowe dane aplikacji oraz wykonywane na nich operacje.

app – warstwa aplikacji. Zawiera implementacje usług, które odpowiadają za realizację przypadków użycia bądź operacji wchodzących w ich skład.

port – warstwa portów. Port to pojęcie elastyczne, nie istnieje ścisła jego definicja. Możemy założyć, że port to interfejs, API, za pomocą którego kontekst ograniczony komunikuje się z infrastrukturą, czyli światem zewnętrznym.

infra – warstwa infrastruktury, czyli implementacja portów. Przykładem może być implementacja repozytorium, które odpowiada za utrwalanie dokumentów (obiektów domeny) w bazie danych. Repozytorium implementuje interfejs, czyli API zdefiniowane przez port.

ui – warstwa interfejsu użytkownika. W naszym przypadku są to komponenty odpowiedzialne za działanie stron JSF.

api – opcjonalna warstwa reprezentująca usługi dostępne przez REST API.

4.5.5. UI

Duża część pracy programistycznej związanej z rozwojem naszego systemu polega na tworzeniu interfejsu użytkownika. Po wielu analizach i szacunkach postanowiliśmy, że w tym obszarze nie decydujemy się na razie na głębsze zmiany. Oczywiście na rynku aplikacji webowych od pewnego czasu zaczyna dominować nowy sposób realizacji interfejsu użytkownika (najczęściej jest to jedno rozwiązanie z trójki: Angular[21], React[22] oraz Vue[23]), ale w naszym przypadku nie istnieje w tej chwili paląca potrzeba takiej zmiany.

Zostajemy przy technologii Java Server Faces, ponieważ dysponujemy dużą biblioteką własnych, dojrzałych i sprawdzonych komponentów, programiści potrafią się nimi posługiwać, a użytkownicy potrafią ich używać. Zmiana w tym obszarze nie stanowi w tej chwili bezpośredniej korzyści, a ryzyko z nią związane istnieje. Co więcej, charakter systemu OSF kłóci się z niezwykle modną ostatnio, proponowaną przez wymienione wyżej biblioteki, ideą „jednostronicowych aplikacji internetowych" (ang. *Single Page Application*). Rozmiar systemu OSF jest bardzo duży, liczba obsługiwanych wniosków również i pomysł, by przeglądarka ładowała to wszystko na starcie jest niewykonalny i nierozsądny. Chcąc korzystać np. z biblioteki Angular i tak całość interfejsu użytkownika trzeba by dzielić na mniejsze części, tworząc zbiór wielu osobnych aplikacji jednostronicowych. W tej sytuacji pozostanie przy sprawdzonej technologii JSF wydaje się wyborem po prostu bardziej rozsądnym. Co oczywiście nie oznacza, że do wymiany interfejsu użytkownika w przyszłości nie dojdzie.

[21] angular.io (dostęp: 17.08.2021)
[22] reactjs.org (dostęp: 17.08.2021)
[23] vuejs.org (dostęp: 17.08.2021)

ROZDZIAŁ 5

ZAKOŃCZENIE

OSF to duży, ciągle rozwijany system informatyczny funkcjonujący na rynku od ponad piętnastu lat. Niniejsza monografia, ze względu na swą ograniczoną objętość, nie jest w stanie w sposób kompletny zaprezentować wszystkich zagadnień dotyczących aplikacji. Nie taki jest też cel monografii. W niniejszej pracy chcieliśmy zaprezentować kilka istotnych aspektów systemu. Ważnych, ale nie jedynych.

Na początku dokonaliśmy prezentacji systemu, opisaliśmy historię jego powstania i przedstawiliśmy podstawowe zadania, które ma realizować. Zadania te, czyli procesy biznesowe obsługiwane przez system OSF wzięliśmy pod lupę w rozdziale drugim. Tam znajduje się bardziej szczegółowy opis funkcji, które system oferuje trzem głównym interesariuszom, czyli Ministerstwu Edukacji i Nauki (MEiN), Narodowemu Centrum Nauki (NCN) i Narodowemu Centrum Badań i Rozwoju (NCBR).

W kolejnych dwóch rozdziałach prezentujemy system z innej perspektywy. Po pokazaniu jego możliwości koncentrujemy się na tym, jak aplikacja powstaje. Przyglądamy się sposobom, w jaki zorganizowany jest proces wytwórczy, jaka jest struktura zespołu, w jaki sposób dodawane są do aplikacji nowe funkcje i jak modyfikujemy istniejące. Bardzo cenna jest też wiedza, jaka była droga dojścia do obecnego, stabilnego i mocno sformalizowanego stanu. Historia systemu zaczęła się od trzyosobowego, bardzo elastycznego jeśli chodzi o podział funkcji zespołu, który finalnie zmienił się w profesjonalną, precyzyjnie funkcjonującą strukturę. Rozdział, w którym opisujemy aktualny sposób jej funkcjonowania wydaje się bardzo cenny dla każdej interesującej się organizacją podobnych przedsięwzięć osoby.

Ostatni, czwarty rozdział to opis ewolucji architektury systemu. Koncentrujemy się w nim nie tylko na szczegółach dotyczących technologii i narzędzi, choć oczywiście staramy się pokazać je dość dokładnie. Ale ten rozdział to przede wszystkim historia podejmowania wielu trudnych decyzji, które pozwoliły zapewnić ciągłe, niezawodne działanie i rozwój systemu w bardzo zmiennym, szybko rozwijającym się otoczeniu. To historia postępu, który dokonał się w OSF także z punktu widzenia architekta i technologa, ale także historia przemyślanych wyborów. Wartościowy jest naszym zdaniem opis dekompozycji monolitu i zastąpienie go zbiorem mniejszych, współpracujących ze sobą podsystemów. Mamy więc nadzieję, że także dla Czytelnika o technicznych zainteresowaniach niniejsza praca będzie cennym źródłem wiedzy i materiałem do przemyśleń.

System cały czas działa i będzie rozwijany. Na pewno w dużym stopniu rozwój ten będzie zgodny z ogólnymi trendami obowiązującymi na rynku IT. Na pewno też wszystkie kolejne, podejmowane przez nas decyzje będą wyważone i poprzedzone głębo-

ką analizą realnych korzyści i ewentualnych strat. Ale dalszego kroczenia drogą zmierzającą do dekompozycji i coraz większej interoperacyjności nie da się uniknąć. Istnieją także pewne rezerwy w obszarze możliwej automatyzacji samego procesu wytwórczego i poprawienia wydajności pracy. Te tematy są dla nas równie ważne, jak kwestie czysto technologiczne. Jedno jest pewne: zespół OSF nie powiedział jeszcze ostatniego słowa.

SPIS ILUSTRACJI

2.1. Główne procesy biznesowe realizowane w systemie OSF 13
2.2. Etapy procesu wnioskowania 14
2.3. Etapy procesu oceny wniosków w Narodowym Centrum Nauki 20
2.4. Etapy procesu oceny wniosków w Ministerstwie Edukacji i Nauki 22
2.5. Etapy procesu oceny wniosków w Narodowym Centrum Badań i Rozwoju 24
2.6. Etapy procesu obsługi finansowania w Narodowym Centrum Badań i Rozwoju 27
2.7. Etapy procesu obsługi finansowania w Ministerstwie Edukacji i Nauki – warianty 27
2.8. Etapy procesu obsługi finansowania w Narodowym Centrum Nauki 28
3.1. Przebieg podzadania programistycznego 37
4.1. Zasada działania frameworka Apache Struts 46
4.2. Liczba kierunków Informatyka według poziomu i formy studiów 52
4.3. Udział kierunków Informatyka wśród kierunków inżynieryjno-technicznych w kolejnych latach według poziomu i formy studiów 52
4.4. Liczba studiowań na kierunkach Informatyka w kolejnych latach według poziomu i formy studiów 53
4.5. Udział studiowań na kierunkach Informatyka wśród studiowań na kierunkach inżynieryjno-technicznych według poziomu i formy kształcenia 53
4.6. Udział studiowań na kierunkach Informatyka w ogóle studiowań według poziomu i formy kształcenia 54
4.7. Liczba absolwentów kierunków Informatyka w kolejnych latach według poziomu i formy studiów 55
4.8. Średnie zarobki ogółem absolwentów kierunków Informatyka w kolejnych latach po dyplomie według poziomu studiów 56
4.9. Względny wskaźnik zarobków absolwentów kierunków Informatyka w kolejnych latach po dyplomie 57
4.10. Względny wskaźnik bezrobocia absolwentów kierunków Informatyka w kolejnych latach po dyplomie 58
4.11. Architektura sześciokątna 69

BIBLIOGRAFIA

[1] A. Barda, J. Söderqvista, *Netokracja. Nowa elita władzy i życie po kapitalizmie*, Wydawnictwa Akademickie i Profesjonalne, Warszawa, 2006.

[2] *Barometr Zawodów*, barometrzawodow.pl (dostęp: 17.08.2021).

[3] E. Bendyk, *Ideologia społeczeństwa informacyjnego*, calculemus.org/lect/mes99-00/spin/1bendyk.html (dostęp: 17.08.2021).

[4] S. Bodoff, E. Armstrong, J. Ball, D. Carson, *J2EE. Vademecum profesjonalisty, wydanie II*, Helion, 2005.

[5] G. Booch, I. Jacobson, J. Rumbaugh, *UML Przewodnik użytkownika*, Wydawnictwa Naukowo-Techniczne, 2012.

[6] F. P. Brooks, K. E. Iverson, *Automatic Data Processing: System/360 Edition*, Wiley, 1969, ISBN 978-0471106050.

[7] M. Castells, P. Himanen, *The Information Society and the Welfare State: The Finnish Model*, Oxford University Press, 2002.

[8] N. Conklin, *3 powody, dla których warto budować systemy monolityczne*, bulldogjob.pl/news/588-3-powody-dla-ktorych-warto-budowac-systemy-monolityczne (dostęp: 17.08.2021).

[9] N. Conklin, *Modular Monolith – modularność drogą do mikroserwisów*, bulldogjob.pl/news/1090-modular-monolith-modularnosc-droga-do-mikroserwisow (dostęp: 17.08.2021).

[10] R. Florida, *Narodziny klasy kreatywnej oraz jej wpływ na przeobrażenia w charakterze pracy, wypoczynku, społeczeństwa i życia codziennego*, Kultura się liczy!, Narodowe Centrum Kultury, Warszawa, 2010.

[11] K. D. Foote, *A Brief History of Microservices*, dataversity.net/a-brief-history-of-microservices (dostęp: 17.08.2021).

[12] M. Fowler, *Microservices, a definition of this new architectural term*, martinfowler.com/articles/microservices.html (dostęp: 17.08.2021).

[13] R. Johnson, J. Hoeller, T. Risberg, C. Sampaleanu, *Spring Framework. Profesjonalne tworzenie oprogramowania w Javie*, Helion, 2006.

[14] B. Jung, *Destruktywny wpływ informatyków na rynek pracy z perspektywy nowych zjawisk na rynku pracy zawodów kreatywnych*, *Roczniki Kolegium Analiz Ekonomicznych*, 2016, 40, s. 83–94.

[15] P. Kieraciński, *Nauka odżyła pod władzą KBN. Rozmowa z prof. Andrzejem Hrynkiewiczem, fizykiem, członkiem Centralnej Komisji ds. Tytułu Naukowego i Stopni Naukowych*, *Forum Akademickie*, 1998, 11/98, forumakademickie.pl/fa-archiwum/archiwum/98/11/artykuly/06-nauka.htm (dostęp: 17.08.2021).

[16] J. Kozłowski, *Polityka naukowa w Polsce – dziedzictwo, stan obecny, perspektywy*, *Nauka i Szkolnictwo Wyższe*, 1997, 1 (9), s. 14–47.

[17] R. C. Martin, *The Single Responsibility Principle*, blog.cleancoder.com/uncle-bob/2014/05/08/SingleReponsibilityPrinciple.html (dostęp: 17.08.2021).

[18] R. C. Martin, *Czysta architektura. Struktura i design oprogramowania. Przewodnik dla profesjonalistów*, Helion, 2018.

[19] Ministerstwo Rozwoju, Pracy i Technologii, *Zawody deficytowe i nadwyżkowe w Polsce. Informacja sygnalna za I półrocze 2019 roku*, dane.gov.pl/pl/dataset/681/resource/18942 (dostęp: 17.08.2021).

[20] Ministerstwo Rozwoju, Pracy i Technologii, *Zawody deficytowe i nadwyżkowe w Polsce. Informacja sygnalna za II półrocze 2019 roku*, dane.gov.pl/pl/dataset/681/resource/22487 (dostęp: 17.08.2021).

[21] S. Newman, *Building Microservices*, O'Reilly and Associates, 2015.

[22] *Ogólnopolski system monitorowania Ekonomicznych Losów Absolwentów szkół wyższych (VI edycja) [dane agregowane na poziomie kierunków]*, ela.nauka.gov.pl/pl (dostęp: 17.08.2021).

[23] *POL-on – zintegrowany system informacji o nauce polskiej i szkolnictwie wyższym*, polon.nauka.gov.pl/siec-polon (dostęp: 17.08.2021).

[24] *Presidency Conclusions, Lisbon European Council, 23 and 24 March 2000*, consilium.europa.eu/uedocs/cms_data/docs/pressdata/en/ec/00100-r1.en0.htm (dostęp: 17.08.2021).

[25] P. Pyzik, *Absencja, fluktuacja i efektywność w branży IT – Rynek Pracy*, rynekpracy.pl/artykuly/absencja-fluktuacja-i-efektywnosc-w-branzy-it (dostęp: 17.08.2021).

[26] D. Rahman, *Comparison of Java web application frameworks*, Rozprawa doktorska, 2019, researchgate.net/publication/330854496_Comparison_of_Java_-web_application_frameworks (dostęp: 17.08.2021).

[27] *Raport z badania „Perspektywy rozwoju polskiej branży ICT do roku 2025" przeprowadzonego przez INVESTIN w 2017 roku na zlecenie Polskiej Agencji Rozwoju Przedsiębiorczości, Warszawa 2017*, parp.gov.pl/component/publications/publication/perspektywy-rozwoju-branzy-ict-do-roku-2025 (dostęp: 17.08.2021).

[28] *Report on Europe and the Global Information Society: Recommendations of the High-level Group on the Information Society to the Corfu European Council*, aei.pitt.edu/id/eprint/1199 (dostęp: 17.08.2021).

[29] *Rozporządzenie Prezesa Rady Ministrów z dnia 18 stycznia 2011 r. w sprawie instrukcji kancelaryjnej, jednolitych rzeczowych wykazów akt oraz instrukcji w sprawie organizacji i zakresu działania archiwów zakładowych*, Dz.U. 2011 nr 14 poz. 67.

[30] S. Sharma, *Mastering Microservices with Java*, Packt Publishing, 2016.

[31] A. Skórzyńska, *Kultura na czas kryzysu. Niedopowiedziany projekt polityczny*, *Kultura Współczesna*, 2011, 5(71), s. 145–163.

[32] J. Spolsky, *Things You Should Never Do, Part I*, joelonsoftware.com/2000/04/06/things-you-should-never-do-part-i (dostęp: 17.08.2021).

[33] *Społeczeństwo informacyjne w Polsce 2020 r.*, Źródło danych GUS, stat.gov.pl/obszary-tematyczne/nauka-i-technika-spoleczenstwo-informacyjne/spoleczenstwo-informacyjne/spoleczenstwo-informacyjne-w-polsce-w-2020-roku,2,10.html (dostęp: 17.08.2021).

[34] *Stateful vs stateless*, redhat.com/en/topics/cloud-native-apps/stateful-vs-stateless (dostęp: 17.08.2021).

[35] *Ustawa z dnia 12 września 1990 r. o szkolnictwie wyższym*, Dz.U. 1990, nr 65, poz. 385.

[36] *Ustawa z dnia 12 stycznia 1991 r. o Komitecie Badań Naukowych*, Dz.U. 1991, nr 8, poz. 1128.

[37] *Ustawa z dnia 22 lutego 1991 r. o zmianie ustawy o jednostkach badawczo-rozwojowych*, Dz.U. 1991, nr 19, poz. 81.

[38] *Ustawa z dnia 8 października 2004 r. o zasadach finansowania nauki*, Dz.U. 2004, nr 238, poz. 2390.

[39] V. Vernon, *DDD dla architektów oprogramowania*, Helion, 2016.

[40] V. Vernon, *DDD. Kompendium wiedzy*, Helion, 2018.