dr hab. Stanisław Ambroszkiewicz
prof. nadzw. IPI PAN

Warszawa, 14 września 2019

Recenzja rozprawy doktorskiej mgr inż. Joanny Kosińskiej pt. : *„Autonomiczne zarządzanie natywnymi aplikacjami w chmurze"*

wykonana dla Rady Wydziału Informatyki, Elektroniki i Telekomunikacji Akademii Górniczo-Hutniczej w Krakowie

1. Jakie zagadnienie naukowe jest rozpatrzone w pracy /teza rozprawy/ i czy zostało ono dostatecznie jasno sformułowane przez autora? Jaki charakter ma rozprawa (teoretyczny, doświadczalny, inny)?

Tematyka rozprawy dotyczy, jak sam tytuł wskazuje, zarządzania aplikacją stworzona według architektury cloud-native i uruchomionej w chmurze obliczeniowej. Korzyści uruchomienia takiej aplikacji w chmurze to między innymi bezpieczeństwo, audytowalność, wysoka dostępność usług, elastyczność, skalowalność, łatwe w użyciu narzędzia do zarządzania takimi aplikacjami.

Wszystko to ma swoją cenę, nie tylko dosłownie. Rzecz w tym, że taka aplikacja musi być odpowiednio zaprojektowana i zrealizowana, tak żeby można było wykorzystać te cechy i te możliwości zarządzania. W rozprawie, proponuje się nawet więcej, tj. autonomiczne i automatyczne zarządzanie taką aplikacją na podstawie z góry ustalonych polityk wyrażonych w sposób deklaratywny.

Z grubsza, architektura aplikacji cloud-native jest oparta na mikro-usługach i kontenerach dla nich uruchamiania. Te mikro-usługi są dynamicznie komponowane (orkiestrowane) w trakcie wykonania aplikacji. W terminologi angielskiej ta architektura sprowadza się do następujących pojęć: *„Continuous delivery, Containers, Dynamic Orchestration, and Microservices"*. Ma to zapewnić elastyczność, skalowalność i niezależność od środowiska wykonawczego. W takich aplikacjach kluczowe są wbudowane mechanizmy, które podnoszą aplikacje z awarii. Jest to osobne ważne zagadnienie.

Doktorantka podejmuje bardzo ważny obecnie problem w technologiach informacyjnych. Jak na doktorat, jest to bardzo trudne wyzwanie.

Zagadnienie naukowe zostało przedstawione i sformułowane przez Autorkę w podrozdziale 1.1 w postaci następującej tezy:
„Cloud-native environments enriched with declarative policy-based approach compliant with components' observability is an effective method of autonomic management realization"

Rozdział 1 (Wstęp) jest dobrym (chociaż zwięzłym) wprowadzeniem do tematyki wyjaśniającym dostatecznie jasno pojęci i terminy użyte w powyższej tezie.

W zakres autonomicznego zarządzania ma wychodzić: instalacja, konfiguracja, optymalizacja i utrzymanie aplikacji, na podstawie deklaratywnych polityk.

Obserwowalność (monitorowanie) stanów aplikacji jest kluczowa. Również kluczowe są operacje, które mogą te stany zmieniać. Wszystko to razem powinno być zapewnione w aplikacji cloud-native przez jej producenta. W rozprawie zastał przestawiony przykładowy (ale też i dosyć wszechstronny) zestaw obserwowalnych cech składający się na stan aplikacji.

Rozprawę można uznać za bardzo dobre studium metodologiczne dotyczące modelowania autonomicznego zarządzania aplikacją cloud-native w chmurze. Jest to z jednej strony teoretyczne podejście przy opracowaniu modelu, ale też i praktyczne podejście (tzw. „proof of concept"), poprzez konkretną implementację tego modelu w celu weryfikacji wymagań stawianych temu modelowi w tezie rozprawy.

2. Czy w rozprawie przeprowadzono w sposób właściwy analizę źródeł / w tym literatury światowej, stanu wiedzy i zastosowań w przemyśle / świadczący o dostatecznej wiedzy autora? Czy wnioski z przeglądu źródeł sformułowano w sposób jasny i przekonywający?

Rozdział 2, zawiera przegląd podstawowych pojęć dotyczących chmury obliczeniowej, aplikacji typu cloud-native, oraz Autonomic Computing. Te ostatnie pojęcie jest kluczowe dla rozprawy, i opiera się na modelu referencyjnym zaproponowanym przez IBM (jakiś czas temu) pod nazwą MAPE-K. Przy czym poszczególne litery tego skrótu odnoszą się do monitorowania, analizy, planowania, wykonania (execution) oraz do wiedzy (knowledge). Istotne są tutaj sensory służące do obserwowania (częściowego) stanu aplikacji podczas jej działania, oraz aktuatory do zmieniania tego stanu poprzez dostępne operacje. Zarówno obserwowalność pewnych cech stanu, jak i operacje do zmieniania tych cech (tj. stanu) muszą być uwzględnione w całym cyklu produkcyjnym aplikacji, począwszy od projektowania, implementacji oraz wdrażania.

Przegląd ten mógłby dotyczyć nie tylko koncepcji i abstrakcji bezpośrednio związanych z tematyką i wynikami rozprawy, ale dotyczyć również konkretnych istniejących narzędzi i platform do zarządzania zasobami w chmurze, takimi jak Kubernetes, Docker, Microsoft Azure Service Fabric, Mesos and Swarm, Cloud Foundry, OpenShift, czy OpenStack. Jest to przecież rozprawa doktorska.

Nie mniej przegląd ten jest wystarczający do zrozumienia na czym polegają wyniki przedstawione w rozprawie.

3. Czy autor rozwiązał postawione zagadnienia, czy użył właściwej do tego metody i czy przyjęte założenia są uzasadnione?

Zasadnicze wyniki rozprawy mieszczą się w Rozdziale 3 oraz Rozdziale 4. Pozostałe rozdziały to kolejno: implementacja systemu, jego ewaluacja, oraz krótkie podsumowanie. Dalej jest dodatek, w którym są wylistowane pliki służące do konfiguracji AMoCNA (autorskiego systemu do autonomicznego zarządzania aplikacjami cloud-native) oraz istotne fragmenty kodu z implementacji. Na końcu jest obszerna bibliografia, jednak nie jest ona uporządkowana alfabetycznie według autorów, co utrudnia czytanie rozprawy. Powinien być użyty BibTeX przy składaniu rozprawy.

W Rozdziale 3, przedstawiona została autorska koncepcja modułu (jako osobnej aplikacji) do autonomicznego zarządzania aplikacjami cloud-native. Na bazie klasycznego i standardowego modelu aplikacji cloud-native w chmurze obliczeniowej, Autorka dokonała prostej ale istotnej abstrakcji wydzielając następujące wzajemnie powiązane pod-moduły: pierwszy jest odpowiedzialny za obserwowanie stanów zarządzanej aplikacji, drugi podmoduł zawiera deklaratywne polityki , oraz trzeci podmoduł decyzyjny odpowiedzialny za kontrolę stanu poprzez realizację reguł, tj. wykonywanie akcji na podstawie obserwowanych stanów aplikacji mających przynieść pożądany efekt zdefiniowany w przyjętej polityce.

Konkretyzacją powyższej abstrakcji jest koncepcja cloud-native MRE-K loop (procesu zarządzania przestawionego w postać pętli) na postawie IBM-owskiej MAPE-K loop.

Istotna i kluczowa jest tutaj zarówno obserwowalność stanu wykonywanej aplikacji, jak również zestaw akcji do kontroli tych stanów.
Pełny zestaw obserwowalnych cech (Observability stack) dotyczy: zasobów (obliczeniowych, pamięciowych i sieciowych) infrastruktury chmury, klastrów kontenerów i orkiestracji w nich, samych mikro-usług uruchamianych w kontenerach, oraz stanów samej aplikacji, które muszą być wyspecyfikowane i dostępne przez samego producenta aplikacji. Formalna specyfikacja powyższych cech jest sama w sobie olbrzymim problemem nie rozwiązanym do tej pory.
Zazwyczaj ten zestaw jest mocno ograniczony i sprowadza się do metryk dostarczanych przez narzędzia takie jak Kubernetes czy Docker, oraz prostych metryk i typów zdarzeń dostarczanych przez producenta aplikacji cloud-native.
To samo dotyczy akcji do kontroli stanu wykonywanej aplikacji. Tak też przyjęła Autorka przy implementacji swojego systemu.

Koncepcja cloud-native MRE-K loop jest realizowana w systemie AMoCNA (tj. Autonomic Management of Cloud-native Applications).

Implementacja systemu AMoCNA jest oparta na OpenStack oraz Kubernetes wraz w towarzyszącymi im narzędziami. Mikro-usługi realizujące cloud-native MRE-K loop są w kontenerach w klastrze Kubernetes i podlegają tam orkiestracji. W tym samym klastrze są też kontenery aplikacji cloud-native, która ma być zarządzana przez AMoCNA. Środowisko wykonania aplikacji w chmurze obliczeniowej jest zarządzane przez OpenStack.

Metryki z Kubernetes wyznaczają obserwowalne cechy zarządzanej aplikacji. Stąd też AMoCNA, z konieczności, dziedziczy te metryki. Wydaje się, że nie może wyjść poza te metryki. Owszem, te metryki są dosyć uniwersalne, ale jednak ustalone i ograniczone, i nie

mogą siłą rzeczy obejmować wszystkich (sensownych i ważnych) aspektów zarządzania aplikacją cloud-native.

Należy tutaj postawić następujące pytanie. Na ile AMoCNA jest uniwersalna w stosunku do zarządzanych aplikacji cloud-native. Jeśli przyjmiemy, że obserwowalność i akcje sprowadzają się do tych zapewnionych w Kubernetes, to odpowiedź jest TAK. Ale przecież konkretna aplikacja cloud-native może posiadać własne specyficzne metryki i typy zdarzeń, jak również własny zestaw akcji do kontroli. Wtedy uniwersalność nie może być zapewniona przez AMoCNA, i nie tylko przez ten system.
Na obronie oczekuję dyskusji na ten temat.

Skoro OpenStack oraz Kubernetes są użyte w implementacji, to powtórzę uwagę dotyczącą przeglądu (Rozdział 2), że wyczerpujący opis tych technologii (i nie tylko tych szczególnych) powinien znaleźć się w rozprawie.

4. Na czym polega oryginalność rozprawy, co stanowi samodzielny i oryginalny dorobek autora, jaka jest pozycja rozprawy w stosunku do stanu wiedzy czy poziomu techniki reprezentowanych przez literaturę światową?

Koncepcja Cloud-native MRE-K loop oparta na deklaratywnych politykach jest oryginalna. Również architektura systemu AMoCNA jest interesująca pod względem technologii informacyjnych i stanowi oryginalny wkład Doktorantki.

Deklaratywne podejście do wyrażania polityk i SLA , oraz wykorzystanie koncepcji AC i dedykowany (czy uniwersalny?) moduł Cloud-native MRE-K loop, realizowany nad aplikacją cloud-native, jest ciekawym rozwiązaniem zarówno teoretycznym jak i technologicznym. Czy może być on uniwersalny i na jakim poziomie, i przy jakich założeniach?

Jeśli w aplikacji cloud-native podczas jej projektowania i implementacji umożliwiono jedynie obserwowalność wydajności i akcje do sterowania tą wydajnością poprzez zwiększenie lub zwalnianie (według aktualnego zapotrzebowania) zasobów (pamięci, kontenerów i powielanie w nich elementarnych mikrousług, szerokości pasma dostępu do sieci), zaś polityki dotyczą tych aspektów, to odpowiedź na powyższe pytanie jest pozytywna.
Ale są inne ważne aspekty zarządzania jak np. bezpieczeństwo, czy rekonfiguracja specyficzna dla konkretnej aplikacji cloud-native. A tego nie da się zrobić w uniwersalny sposób. Potrzebne są narzędzia do tworzenia API w aplikacjach do zarządzania tymi aspektami. W tym kontekście, całościowe podejście proponowane przez platformę Chef ((https://www.chef.io) wydaje się być odpowiednie.

Wydaje się, że obecnie najbardziej zbliżony, do proponowanego podeścia w rozprawie, jest framework
Chef Habitat (https://www.chef.io/cloud-native/) do zarządzania cloud-native aplikacji. Służy on również do projektowanie, budowania i wdrażania aplikacji typu cloud-native.
Jest to jednak zamknięta platforma. Nie mniej, Doktorantka powinna sie do niej odnieść na obronie.

5. Czy autor wykazał umiejętność poprawnego i przekonywującego przedstawienia uzyskanych przez siebie wyników /zwięzłość, jasność, poprawność redakcyjna rozprawy/ ?

Miejcami tekst jest rozwlekły i jest wiele powtórzeń. Brakuje konkretów, takich jak formalna deklaratywna specyfikacja polityk, SLA, oraz QoS. Można odnieść ogólne wrażenie, że Autorka rozprawy znacznie więcej zrobiła niż zostało to przedstawione w rozprawie. Dotyczy to zwłaszacza kluczowych pojęć opisanych nieformalnie, ale również i technicznych szczegółów implementacji, gdzie Doktorantka swobodnie operuje (bez uprzedniego przedstawienia czy nawet zgrubnego opisu) takimi platformami i narzędziami jak OpenStack czy Kubernetes.

6. Jakie są słabe strony rozprawy i jej główne wady?

Brak formalnej rozszerzalnej uniwersalnej ontologii do opisy stanów aplikacji cloud-native to najsłabsza strona rozprawy.
Przyjęcie UML wiąże się zbezpośrednio z reprezentacją reguł zarządzania dostępnych w aplikacji cloud-native, jak i w samej chmurze, gdzie jest uruchomiona ta aplikacja.
Również UML jest pomocny i użyty do realizacji złożonych procedur do zarządzania.

UML nie jest dobrym językiem do opisu stanów uruchomionej aplikacji i środowiska.
Powinien to być język w pełni delklaratywny opisujące tę ontologię i umożliwiający automatyczne wnioskowanie.

Potrzebna jest więc uniwersalna ontologia i formalny język jej opisu.
Wydaje się, że elementarne usługi do zarządzania najlepiej realizować w postaci mikrousług z dobrze opisanym interfejsem wyrażonym w tym języku. Typ takiej operacji to typ akcji (operacji) jako tzw. *pre-condition* ora *effec*t, tj. jako para formuł *(formuła opisująca stany początkowe, formuła opisująca stany końcowe)*.
Polityki, w tym QoS, wyrażone w tym języku sprowadzają się do opisu zbioru pożądanych stanów, być może z preferencjami, które z nich są lepsze i na ile.

Jest szereg nurtów badań nad takimi ontologiami oraz publikacji im poświęconych, np.:

Bergmayr, Alexander, Uwe Breitenbücher, Nicolas Ferry, Alessandro Rossini, Arnor Solberg, Manuel Wimmer, Gerti Kappel, and Frank Leymann. "A systematic review of cloud modeling languages." ACM Computing Surveys (CSUR) 51, no. 1 (2018): 22.

Bergmayr, A., Rossini, A., Ferry, N., Horn, G., Orue-Echevarria, L., Solberg, A., & Wimmer, M. (2015, September). The evolution of CloudML and its manifestations. In Proceedings of the 3rd International Workshop on Model-Driven Engineering on and for the Cloud (CloudMDE) (pp. 1-6). http://ceur-ws.org/Vol-1563/paper3.pdf

Bassiliades, Nick, Moisis Symeonidis, Panagiotis Gouvas, Efstratios Kontopoulos, Georgios Meditskos, and Ioannis Vlahavas. "PaaSport semantic model: An ontology for a platform-as-a-service semantically interoperable marketplace." Data & Knowledge Engineering 113 (2018): 81-115.

Sugeruję Autorce zapoznanie się (może się to przydać na obronie) również z pracą doktorską o podobnej tematyce, tj.
An integrated modeling framework for managing the deployment and operation of cloud applications. PhD thesis
M Hamdaqa - 2016 - uwspace.uwaterloo.ca

Można się zastanowić, czy w autonomiczny zarządzaniu nie należy dodać jeszcze jednej abstrakcyjnej warstwy opartej na otwartej i rozszerzalnej ontologii dziedzinowej. Stany aplikacji można by opisywać w tej ontologii. Również typy akcji (widziane jako usługi w paradygmacie SOA) do zarządzania, można by opisywać w tej ontologii. Dodatkowo możliwe by było wnioskowanie oraz planowanie wykonywania sekwencji akcji. Polityki, QoS oraz SLA można by również reprezentować deklaratywnie w języku ontrologii, jako intencje i cele, tj. formuły opisujące podzbiory stanów.
Ta dodatkowa abstrakcyjna warstwa byłaby deklaratywna, symboliczna i oparta na formalnym wnioskowaniu i planowaniu. Pod nią powinna być warstwa konkretyzacji, w której, do wybranego planu abstrakcyjnego (w postaci sekwencji typów akcji), przypisywane byłyby konkretne akcje. Następna warstwa to wykonanie konkretnego planu wraz z mechanizmami adaptacji do nieprzewidzianych zdarzeń i reagowania na niepowodzenia.

7. Jaka jest przydatność rozprawy dla nauk technicznych?

Potencjalnie bardzo duża. Wskazane jest aby prace były kontynuowane w celu zaproponowania formalnego systemu reprezentacji wiedzy (w tym ontologii) o stanie autonomicznych aplikacji w chumrach, oraz reguł wnioskowania oraz deklaratywnej kontroli, adaptacji, i ogólnie zarządzania takimi aplikacjami w trakcie ich działania. Następnie, warto podjąć próbę standaryzacji języka, ontologii, reguł wnioskowania i deklaratywnych procedrr zarządzania.

W konkluzji należy jeszcze raz podkreślić ambitną próbę Doktorantki zmierzenia się z bardzo trudnym naukowym i technlogicznym wezwaniem. Wyniki opisane w rozprawie można uznać za ciekawe studium metodologiczne nad bardzo ważnym problemem jakim jest autonomiczne zarządzanie aplikacją typu cloud-native w chmurze obliczeniowej.

Zaproponowane rozwiązanie w posatci Cloud-native MRE-K loop oparte o deklaratywne polityki, i jego implementacja w autorskim systemie AMoCNA, stanowią istotny wkład do badań w tej tak bardzo ważnej współczesnie tematyce dotyczącej zarządzania aplikacjami w chmurze obliczeniowej, ale również nierozłącznie z nim związanym projektowaniem oraz budowaniem i wdrażaniem takich aplikacji.

W konkluzji stwierdzam, że rozprawa doktorska mgr inż. Joanny Kosińskiej <u>spełnia w pełni</u> wymagań stawianych rozprawom doktorskim przez obowiązującą Ustawę o stopniach naukowych i tytule naukowym oraz o stopniach i tytule w zakresie sztuki z dnia 14 marca 2003 roku (Dz.U. Nr 65 poz. 595 z późniejszymi zmianami).

8. Do której z następujących kategorii Recenzent zalicza rozprawę:
a) nie spełniająca wymagań stawianych rozprawom doktorskim przez obowiązujące przepisy
b) wymagająca wprowadzenia poprawek i ponownego recenzowania

c) <u>spełniająca wymagania</u>
d) spełniająca wymagania z wyraźnym nadmiarem
e) wybitnie dobra, zasługująca na wyróżnienie