УДК: 681.3.016

В.В. Ефимов, Д.А. Щемелинин, К.А. Яковлев

Интеграционная модель данных для управления непрерывным обслуживанием глобально распределенных вычислительных систем

Integration Data Model for Continuous Service Delivery in Globally Distributed Computing System

Аннотация. Предложены новые модели данных и подход к интеграции систем мониторинга событий, контроля неполадок, установки обновлений, и прочих процессов, которые обслуживаются разрозненными командами и инструментальными средствами автоматизации в большой глобально распределенной инфраструктуре. Новая интеграционная архитектура и модели управления внедрены в крупной интернациональной IT компании, что позволило обеспечить непрерывность цикла обслуживания и обновления Интернет сервисов в быстро развивающейся сфере телекоммуникаций.

Ключевые слова. Интеграционная модель данных, непрерывный цикл обслуживания, глобально распределенные вычислительные комплексы.

Abstract. New approach and data models are proposed to integrate the monitoring system, incident and problem management, service delivery and change management, analytics and other processes, which are managed by separated operations teams and automation tools in a big globally distributed computing infrastructure. New integration architecture and data models are implemented in the International IT Company, which allowed providing a full cycle of continuous delivery of Internet services in a rapidly growing telecommunications area.

Key words. Integration data model, continuous service delivery, globally distributed computing system.

Введение

В крупной международной IT компании, предоставляющей облачные услуги в сфере телекоммуникаций, существует потребность в обслуживании огромного парка вычислительных ресурсов (физических серверов, виртуальных машин, программного обеспечения [1]), глобально распределенных в удаленных центрах обработки данных на разных континентах. В данной работе использованы реальные сведения об американской компании RingCentral [2], предоставляющей облачные телекоммуникационные услуги в режиме 24/7 в 27 странах Северной Америки, Западной Европы и Юго-Восточной Азии.

Компания обслуживает более 10 тысяч серверов, расположенных в 6 удаленных центрах обработки данных. С каждого удаленного сервера собирается информация в централизованную систему мониторинга событий. Суммарный поток данных составляет более 13 тысяч значений в секунду. Каждое событие анализируется для выявления аномального состояния, например, завышенного расхода ресурсов ЦПУ. Таких аномалий в системе фиксируется и устраняется около 3 тысяч в день.

Аномалия может привести к потере резервирования вычислительных ресурсов, снижению производительности и даже к отключению сервиса и отказу в обслуживании пользователей. В среднем фиксируется 3 таких инцидента в день.

Повторяющиеся инциденты – это проблема, которая требует выявления первопричины и исправления базовой ошибки, после чего система должна быть обновлена для предотвращения повторения инцидента в будущем. Среднее количество изменений в глобально распределенной инфраструктуре компании RingCentral, включая аварийное исправление ошибок и плановое добавление новых возможностей системы, – 30 в день, причем за 2016 год наблюдается рост практически в 2 раза (рис. 1).

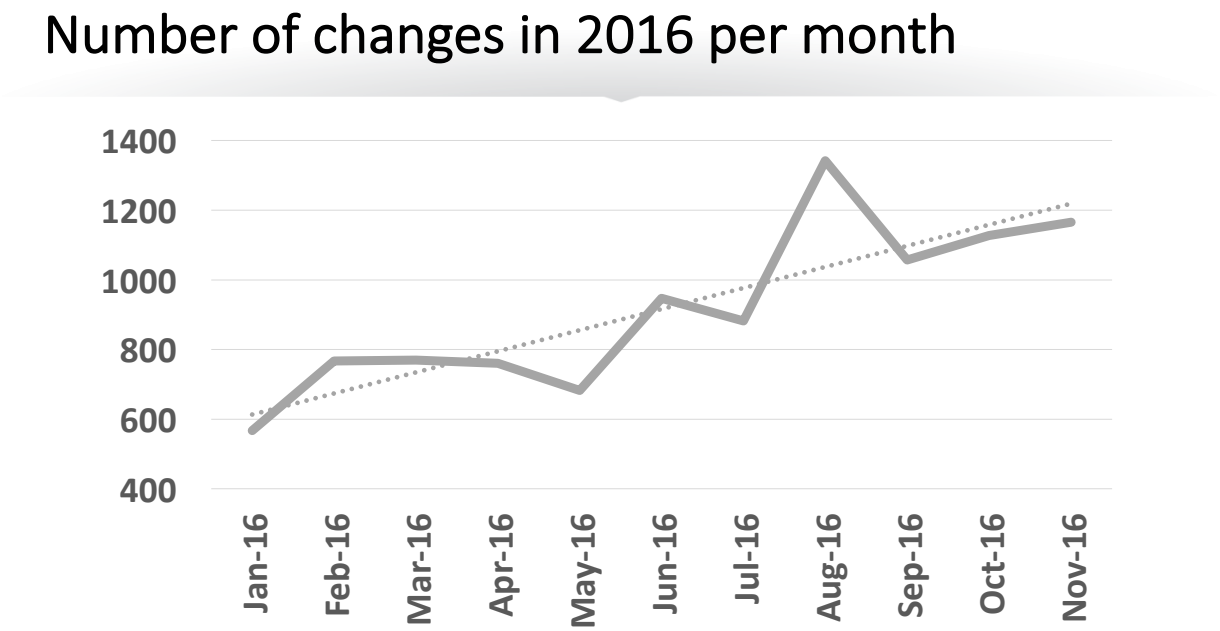

Рис. 1. Количество изменений в месяц в облачной инфраструктуре компании RingCentral в 2016 г.

Учитывая глобальный масштаб всей инфраструктуры, большое количество событий, аномалий и изменений в системе, а также динамику их роста, возникла объективная необходимость повышения эффективности разрозненных процессов эксплуатации посредством их интеграции.

Эксплуатация облачных сервисов

Для эксплуатации облачных сервисов и их восстановления в случае сбоя в IT компаниях традиционно применяются следующие процессы [3, 4]:

1) мониторинг событий – сбор и анализ информации о состоянии удаленных серверов;

2) управление инцидентами – процесс, нацеленный на максимально быстрое восстановление сервисов после сбоя;

3) контроль за проблемами – процесс выявления и исправления ошибок в системе и предотвращения таким образом повторных инцидентов;

4) управление изменениями – плановое обновление системы без нарушения работоспособности сервисов в режиме 24/7.

Каждый из этих процессов обеспечивается отдельными подразделениями эксплуатации, которые разрознены географически и используют разнородные инструментальные средства автоматизации (табл. 1).

Табл. 1. Процессы эксплуатации и средства их автоматизации

Процесс	**Подразделение**	**Средство автоматизации**
Мониторинг событий	Команда мониторинга	Zabbix
Управление инцидентами	Центр эксплуатации сети	Incident Management Portal
Контроль за проблемами	Команда аналитиков и менеджеров	JIRA
Управление изменениями	Команда по внесению изменений	Auto-Deployment System

Анализ всех этих процессов эксплуатации показал:

а) они связаны логически, имея в виду что один процесс может инициировать другой (изменение в системе порождает новые события, события сигнализируют об инциденте, инцидент инициирует процесс анализа проблемы, который в свою очередь заканчивается внесением определенных изменений в систему);

б) процессы связаны с точки зрения данных, задействованных в этих процессах (результаты одного процесса являются исходной информацией для другого, как показано на рис. 2).

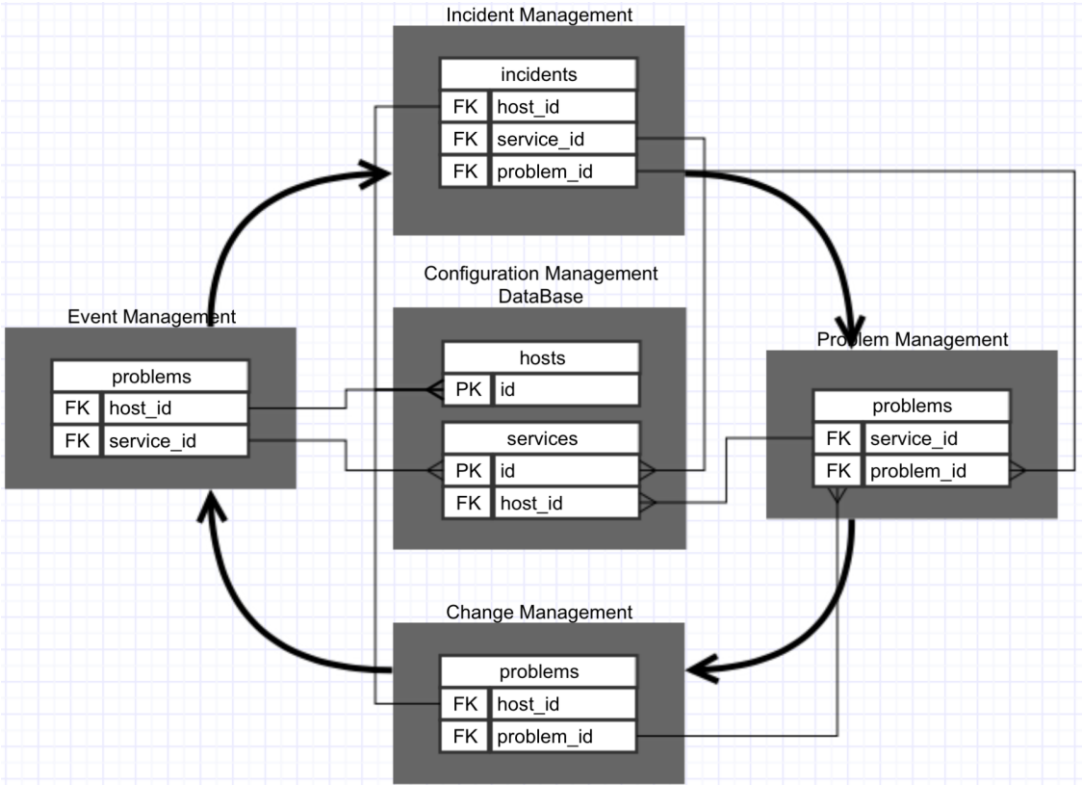

Рис. 2. Взаимосвязь процессов эксплуатации облачных сервисов

Учитывая выявленные взаимосвязи процессов (рис. 2), были сформулированы следующие требования к задаче интеграции с целью повышения эффективности эксплуатации:

1) требуется уметь быстро и надежно определить наиболее подходящее время для внесения изменений в систему с учетом других планируемых обновлений, а также текущего состояния системы – наличия ошибок и проблем;

2) для эффективного контроля за проблемами требуется видеть, на каком этапе находится решение проблемы, включая недавние и текущие инциденты, связанные с данной проблемой, планируемые изменения для устранения причины проблемы;

3) для быстрого и правильного устранения проблемы требуется видеть взаимосвязь между недавними изменениями и текущими аномалиями, чтобы в случае ошибок быстро отменить изменение, вызвавшее отказ.

Интеграционная модель данных

Для реализации поставленной задачи разработана информационная модель интеграции с использованием инструментальных средств моделирования БД Gliffy [5, 6] (рис. 3), в основе которой положена предыдущая работа авторов [7].

Согласно новой интегрированной структуре данных, облачная инфраструктура состоит из системных единиц (System Unit), обладающих определенными свойствами. Каждая системная единица имеет системные связи (System Relation). Системная связь объединяет две системных сущности так, что их класс соответствует типу данной связи. В интеграционной модели данных реализованы следующие реляционные связи с логикой «родитель-потомок»: сетевой интерфейс системной единицы, физическое соединение, сетевой интернет интерфейс, роутинг интерфейс, сервис доменных имен, сервисный интернет интерфейс, сервисное соединение, системное соединение, отображение портов, реверсивное проксирование.

Системный компонент (System Component) является единственным родителем для всех системных единиц, обладающих одинаковыми свойствами и их значениями. Системный компонент также может представлять сущность без дочерних системных единиц – пользовательский агент (User Agent Client). Реализованы следующие свойства системного компонента:

1) сервисная роль: сервисный шлюз, реверсивный прокси, база данных, кэш, ведущий кластера, ведомый кластера, поставщик данных, шлюз партнера и прочие.

2) тип: виртуальная машина, контейнер, сервер, устройство хранения данных, коммутатор, маршрутизатор, источник питания, стойка, центр обработки данных и прочие.

3) Производитель, поставщик.

4) Модель, тип, торговая марка, брэнд.

5) Операционная система, прошивка, версия.

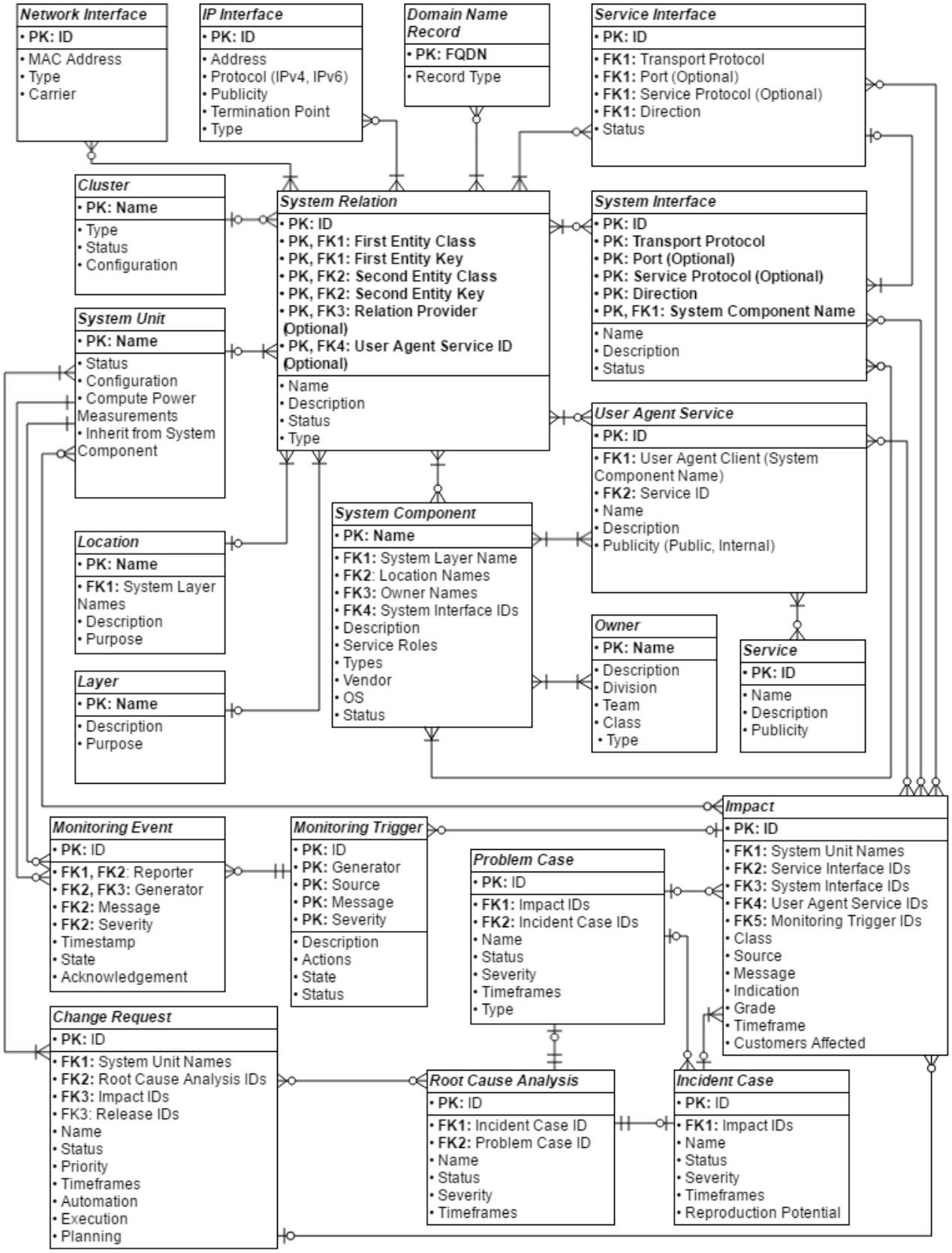

Рис. 3. Интеграционная модель данных

Системный компонент может быть представлен в нескольких экземплярах (Location), имеющих разные предназначения: обслуживание конечных пользователей, тестирование, разработка. Системный компонент представляет один и только один системный уровень (Layer), выполняющий заданную задачу по обработке пользовательских запросов.

Пользовательский сервис (User Agent Service) – это абстрактная сущность, призванная отображать системные связи (сервисное соединение, системное соединение, отображение портов, реверсивное проксирование), используемые для предоставления того или иного сервиса компоненту типа пользовательский агент (User Agent Client).

Системная единица может произвести мониторинговое событие (Monitoring Event) при помощи генератора событий (Generator). У мониторингового события может быть только один генератор (системная единица). Мониторинговый триггер (Monitoring Trigger) – это виртуальная сущность, введенная для отображения мониторинговых событий, которые уже произошли (история мониторинга) и которые могут произойти в будущем (спецификация мониторинга). Мониторинговое событие, представленное мониторинговым триггером и отметкой времени, индицирует воздействие (Impact) на пользовательский сервис, системную единицу, системное или сервисное соединение.

Воздействие также может быть индицировано по пользовательским обращениям. Воздействие имеет временной диапазон и насчитывает следующие градации: прерывание, сокращение качества, обрыв сессий, потеря избыточности, потеря емкости. Воздействие может затрагивать пользователей или нет. Если воздействие не затрагивает пользователя, оно расценивается как потенциальное и должно принадлежать проблеме (Problem Case). Если же воздействие затрагивает пользователей, то оно должно принадлежать инциденту (Incident Case). Если по результатам расследования причин (Root

Cause Analysis) инцидент имеет большую вероятность повторения, он должен стать частью проблемы.

Проблема имеет собственное расследование причин. Результатом расследования причин инцидента является экстренный запрос на изменение (Change Request). Результатом расследования причин проблемы может являться как экстренный, так и запланированный запрос на изменение. В рамках запроса на изменение происходит обновление конфигурации системных единиц. Запрос на изменение может быть вызван не только расследованием причин проблемы или инцидента, вне зависимости от источников изменений они способны вызвать воздействие.

Архитектура интегрированной системы эксплуатации облачных сервисов

Разработанная информационная модель интеграции процессов эксплуатации облачных сервисов реализована на практике и внедрена в глобально распределенной инфраструктуре компании RingCentral, где показала свою эффективность в обеспечении непрерывного цикла обслуживания и обновления облачных сервисов в условиях их быстрого развития и постоянной модернизации.

На рис. 3 приведена архитектура интегрированной системы эксплуатации облачных сервисов компании RingCentral, где реализованы подходы:

1. Использование единого механизма авторизации (single sign on).

2. Микро-сервисная архитектура виртуальных серверов.

3. Использование шины передачи сообщений по принципу оповещения клиентов для обмена данными между микро-сервисами.

4. Доступ к данным через абстракцию API.

Для новой архитектуры разработан интегрированный интерфейс (рис. 4) с целью визуализации всех текущих событий в системе, их корреляции между собой, с последними изменениями и известными хроническими проблемами.

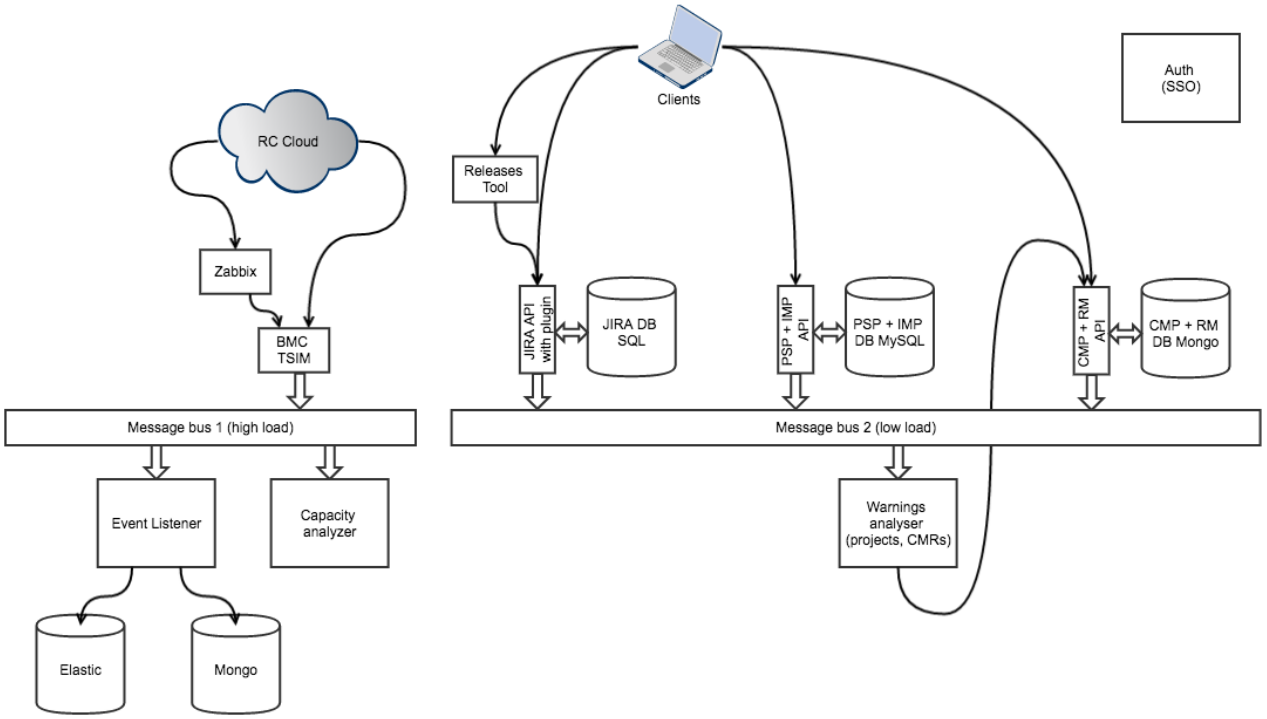

Рис. 3. Архитектура интегрированной системы эксплуатации облачных сервисов компании RingCentral

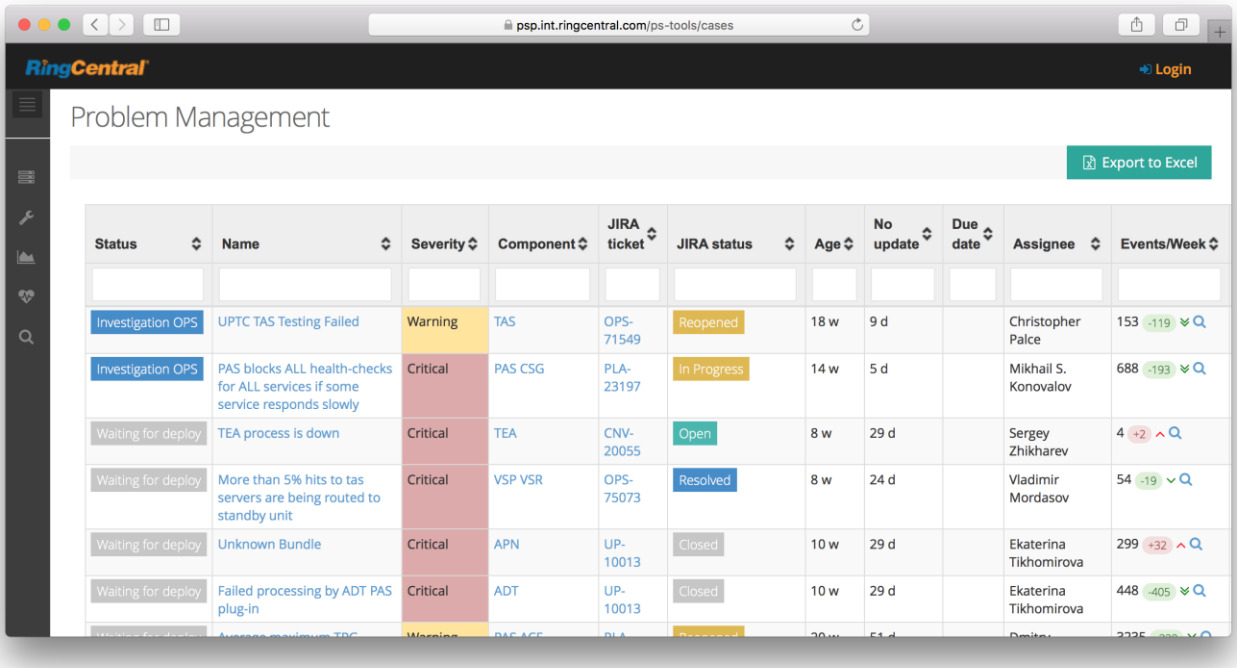

Рис. 4. Интегрированный интерфейс контроля за проблемами, показывающий название проблемы (Name), соответствующую заявку на исправление ошибки (JIRA ticket) и количество таких аномалий за текущую неделю (Events/Week)

Заключение

В данной работе средствами интеграции решается задача повышения эффективности эксплуатации облачных сервисов крупной международной IT компании с глобально распределенной инфраструктурой, большим количеством изменений, аномалий, инцидентов и хронически повторяющихся проблем, а также быстрой динамикой роста данных показателей. Сделан анализ существующих процессов эксплуатации и выявлены взаимосвязи между ними. Разработана новая информационная модель интеграции и приложения визуализации интегрированных данных, которые внедрены в существующую глобально распределенную инфраструктуру компании и эффективно эксплуатируются для управления облачными сервисами.

Дальнейшей проработки требует вопрос интеграции других разрозненных средств автоматизации, используемых в различных процессах установки и тестирования облачных сервисов, которые в настоящее время никак не связаны между собой.

Литература

1. Bernstein D. Containers and Cloud: From LXC to Docker to Kubernetes. IEEE Cloud Computing, Vol. 1, Issue 3, 2014. – http://ieeexplore.ieee.org/document/7036275/

2. RingCentral Inc. – http://ringcentral.com

3. ITIL Service Operations. – https://ru.wikipedia.org/wiki/ITIL

4. Ardulov Y., Shchemelinin D., and Mescheryakov S. Monitoring and Remediation of Cloud Services Based on 4R Approach. Proceedings of the 41st International IT Capacity and Performance Conference by CMG, San Antonio, TX, USA, 2015. – http://www.cmg.org/publications/conference-proceedings/conference-proceedings2015/

5. Lanubile F., Ebert C., Prikladnicki R., and Vizcaino A. Collaboration Tools for Global Software Engineering. IEEE Software, Vol. 27, Issue 2, 2010. – http://ieeexplore.ieee.org/abstract/document/5420797/

6. Мещеряков С.В., Иванов В.М. Методы оптимального проектирования баз данных производственного оборудования. – СПб: Изд-во Политехн. ун-та, 2012. – http://gpupress.ru/

7. Volkov A.N., Efimov V.V., Mescheryakov S.V., Shchemelinin D.A. Integrated Data Model for Managing a Multi-Service Dynamic Infrastructure. Computer Modeling and Simulation: труды междунар. науч.-техн. конф. КОМОД-2014, СПб, Изд-во Политехн. ун-та, 2014. – http://dcn.icc.spbstu.ru/index.php?id=344

Сведения об авторах

Ефимов Вадим Вячеславович
Санкт-Петербургский политехнический университет Петра Великого
Рабочий адрес: 195251, Санкт-Петербург, Политехническая ул., 29
Ученая степень, звание: нет
Должность: аспирант
Электронная почта: 2vadim@inbox.ru

Щемелинин Дмитрий Александрович
Санкт-Петербургский политехнический университет Петра Великого
Рабочий адрес: 195251, Санкт-Петербург, Политехническая ул., 29
Ученая степень, звание: к.т.н.
Должность: докторант
Электронная почта: dshchmel@gmail.com

Яковлев Константин Андреевич
ООО «Дино системс»
Рабочий адрес: 190020, Санкт-Петербург, Старо-Петергофский пр., 19
Ученая степень, звание: нет
Должность: системный аналитик
Электронная почта: iakovlev.ka@gmail.com

Efimov Vadim Vyacheslavovich
Peter the Great St. Petersburg Polytechnic University
Postal address: 29, Politechnicheskaya St., St. Petersburg, 195251, Russia

Shchemelinin Dmitry Alexandrovich
Peter the Great St. Petersburg Polytechnic University
Postal address: 29, Politechnicheskaya St., St. Petersburg, 195251, Russia

Yakovlev Konstantin Andreevich
Dino Systems
Postal address: 19, Staro-Petergofsky Av., St. Petersburg, 190020, Russia