Дослідження доцільності впровадження концепції Internet of Everything в галузь вищої освіти в Україні

Соколенко П. Ю.
науковий керівник: Конрад Т. І.
Кафедра інженерії програмного забезпечення,
Навчально-науковий інститут комп'ютерних інформаційних тхнологій,
Національний авіаційний університет,
Київ, Україна
sokolenko.py@gmail.com

***Анотація* — робота присвячена впровадженню концепції Internet of Everything у галузь вищої освіти в Україні. У роботі запропоновано використання нової концепції з метою покращення процесу навчання та проведення наукових досліджень. Також у роботі розглянуті аспекти впливу складових концепції Internet of Everything на галузь освіти.**

Ключові слова — Internet of Everything, Internet of Thing, освіта, інтернет речей.

I. Вступ

Концепція Internet of Everything(IoE) виникла в наслідок природного розвитку Internet of Thing(IoT), IoE охоплює більш широке поняття зв'язності з точки зору використання сучасних технологій зв'язку. IoE складається з чотирьох ключових елементів:

- люди - розглядаються як кінцеві "вузли", з'єднані через Інтернет для обміну інформацією (наприклад соціальні мережі, датчики здоров'я та інше);
- речі - фізичні датчики, пристрої, виконавчі пристрої та інші елементи, що генерують дані або отримують інформацію з інших джерел (наприклад розумні термостати та гаджети);
- дані - аналітичні дані обробляються та перетворюються в корисну інформацію, що дозволяє прийняти інтелектуальні рішення та виконувати механізми контролю;
- процес - використання зв'язку між даними, речами та людьми для додавання вартості, приклади включають використання пристроїв інтелектуальних та соціальних мереж для реклами відповідних пропозицій охорони здоров'я потенційним клієнтам [1].

II. Постановка проблеми

Розвиток технологій безперечно впливає на усі сфери життя. Сфера освіти не виключення.

У всьому світі IoE в галузі освіти знаходиться на ранніх стадіях розвитку, проте деякі навчальні заклади вже прокладають шлях до використання цієї нової концепції. Тому актуальним питанням є дослідження впровадження Internet of Everything у галузь вищої освіти в Україні з метою модернізації методів організації процесу навчання.

III. Основна частина

Основними складовими IoE, що впливають на процес навчання визначено наступні: люди, дані, речі та процес [2].

Люди. Сьогодні більшість людей мають аккаунти у різних соціальних мережах. Представники сектору освіти мають розуміти, як люди підключаються до Інтернету, щоб вдосконалити вплив на їх навчання. Велике значення має пошук необхідних фахівців для залучення у процес навчання. Кожна людина стає "вузлом" в мережі і їм доведеться підключатися, не тільки до роботи провідних фахівців, а і до однолітків, які мають подібні інтереси. Таким чином, вони будуть ділитися ідеями, обговорювати нові дослідження чи останні події у своїй області навчання.

Поява великих відкритих онлайнових курсів - це ще один крок до глобальної освіти. Деякі провідні світові університети роблять своїх професорів відкритими для людей зі всього світу, а інтернет-форуми - це ще один засіб для розвитку і поєднання людей з усіх сфер життя.

Через систему освіти є можливість знайти експертів та залучити їх до навчання в режимі реального часу або через записане відео. IoE допоможе зв'язати учнів, які знаходяться віддалено, але здатні навчатися та брати участь у навчанні. Крім того, IoE надасть меншинам та учням-інвалідам доступ до високоякісного навчання та рівноправної взаємодії.

IoE також може підтримувати професійний розвиток для викладачів, які можуть затверджувати нові моделі навчання, оскільки дані про їх роботу збираються за допомогою відгуків студентів, досягнень викладачів та відеозаписів. Ці дані можуть бути використані для вивчення сильних та слабких сторін викладачів та є основою для обговорення та подальшої адаптації їх знань для навчального процесу. Висококласні методи викладання можуть бути профілізовані за допомогою записаних відео, які будуть доступними для інших викладачів як інструмент професійного розвитку, тому буде використаним для пояснення та поширення моделей навчання.

Дані. Останнім часом активно розвиваються речі, пов'язані з Інтернетом, вони стають

розумнішими, надаючи більше корисної інформації. Наслідки цього в навчанні є доволі значними. Наприклад, в рамках своїх досліджень студенти можуть обирати фізичні об'єкти, збирати дані про ці об'єкти, а потім передавати цю інформацію іншим програмам для аналізу, підвищуючи точність своїх досліджень.

Дослідження показали, що доступ до інформації в реальному часі та взаємодія з експертами дійсно впливає на навчання. Одним з таких прикладів є Клівлендська клініка у штаті Огайо, Сполучені Штати Америки (США), де біологію людини вивчають у середніх школах за допомогою лапароскопічної хірургії на основі відеоконференцій [3]. Один хірург розповідає про особливості та функції серця та процедуру, а інший проводить операцію. У той же час учні можуть задавати питання. Результатом є посилення мотивації, більше учнів прагнуть стати медсестрами, лікарями або медичними працівниками. Окрім забезпечення точних досліджень і роботи з маніпуляціями з реальними даними, вони можуть також зробити свій внесок у бази даних, стаючи членами експертних спільнот у різних дослідницьких проектах. Вони не тільки контактують із дослідниками, але й працюють з ними, допомагаючи вирішувати місцеві та глобальні проблеми.

У рамках навчальної діяльності, пов'язаної з фізичною культурою, студенти можуть використовувати датчики для контролю за їх повсякденною активністю, збираючи дані про кількість кроків під час ходи та бігу, частоту серцебиття та інші функції метаболізму.

Речі. Речі - це фізичні елементи, які можна підключати як до Інтернету, так і до людей через датчики. Датчики дають речам "голос": шляхом захоплення даних датчики дозволяють речам стати контекстом, надаючи більше експериментальної інформації, яка допомагає людям та машинам приймати відповідні та цінні рішення. Наприклад, сьогодні в мостах використовуються розумні датчики для моніторингу температури, структурної цілісності та щільності руху в режимі реального часу. Таким чином, студенти можуть вивчати фізику, використовуючи свої портативні пристрої для збору та спостереження за мостом при зміні завантаження.

У освіті сенсори з підтримкою IP можуть бути приєднані до артефактів для контролю показників температури, стану або місця розташування об'єкта в режимі реального часу, забезпечуючи постійний потік інформації для археологів або студентів. Це також гарний спосіб для дослідження важкодоступних тварин, тощо.

Датчики також відіграють провідну роль у сфері безпеки студентів. У Окленді, штат Каліфорнія(США), камери безпеки та датчики руху інтегровані в мережу шкіл для моніторингу об'єктів.

Процес. Процес відіграє важливу роль у тому, як люди, дані та речі взаємодіють між собою, щоб забезпечити цінність у світі IoE. При правильній організації процесу зв'язки стають актуальними, а потрібна інформація надходить людині в потрібний час у відповідний спосіб. Забезпечення доступу людей до можливостей навчання, які задовольняють їхні потреби, зробить освіту більш ефективною, оптимізує час на оволодіння інформацією та мотивуватиме студентів. Такі можливості також сприятимуть зацікавленню студентів до процесу пізнання та застосування нових знань, що є життєво важливим для майбутніх успіхів у навчанні та суспільній самореалізації.

Велике значення має зворотний зв'язок щодо успішності студента. Наприклад, студент може спостерігати за своїм рейтингом в режимі реального часу відносно тих, хто навчається на тому ж курсі.

IoE змінить спосіб проведення електронного оцінювання. Майк Ллойд, генеральний директор компанії Edutech Associates, провів дослідження електронної оцінки у системах K-12 [4] у всьому світі та окреслив модель подальших подій. Він описує сценарій, в якому учень підтверджує якість свого навчання через серію електронних оцінок. Щоб отримати повну акредитацію через будь-який офіційний канал, учні можуть отримати доступ до акредитованої екзаменаційної зони, де складатимуть іспити.

IV. ВИСНОВКИ

Досліджено концепцію Internet of Everything та позитивний досвід з її впровадження в США. Представлено основні складові, що впливають на процес навчання в межах даної концепції, серед яких люди, речі, дані та процес. Тому доцільним є впровадження даної концепції до галузі вищої освіти в Україні.

Список використаних джерел

[1] Сайт «IoT TechExpo» [Електронний ресурс]. – Режим доступу: https://www.iottechexpo.com/2016/01/m2m/ioe-vs-iot-vs-m2m-whats-the-difference-and-does-it-matter/

[2] Сайт «CISCO» [Електронний ресурс]. – Режим доступу: https://www.cisco.com/c/dam/assets/sol/ent/day-in-the-life/education/index.html

[3] Сайт «Huffingtonpost» [Електронний ресурс]. – Режим доступу: https://www.huffingtonpost.com/dr-michelle-selinger/ cisco-knowing-everything-when_b_4074269.html

[4] Сайт «Ruabase» [Електронний ресурс]. – Режим доступу: https://rb.ru/story/10-edtech-companies/

Designing Action Language for Foundational UML Metamodel

Teslenko A.V.
Scientific advisor: PhD assoc. Prof. Chebanyuk O.V.
Software Engineering Department
Institute of Computer and Informational Technologies
National Aviation University
Kyiv, Ukraine
anastasiavtes@gmail.com

***Abstract* – this paper illustrates the process of designing metamodel of Action Language for Foundational UML metamodel using Ecore.**

Keywords: Action Language for Foundational UML (ALF), Ecore, Eclipse Modeling Framework (EMF), Model-Driven Architecture (MDA), metamodel, abstraction, transformation, expression, statement, syntax and semantics of modeling language

I. MAIN TERMS

Metamodel – a metamodel is a model of a model. Metamodeling – a modeling process which takes place one level of abstraction and logic higher than the standard modeling process.

ALF – The Action Language for Foundational UML (ALF) is a textual surface representation for UML modeling elements. The Alf specification provides a model for the abstract syntax of the language.

Ecore – the Ecore metamodel is a powerful tool for designing Model-Driven Architecture, the core metamodel at the heart of EMF. Is also defined in terms of itself. It allows expressing other models by leveraging its constructs.

EMF – The core Eclipse Modeling Framework (EMF) includes a metamodel (Ecore) for describing models and runtime support for the models including change notification, persistence support with default XMI serialization, and a very efficient reflective API for manipulating EMF objects generically.

II. PROBLEM FORMULATION

In software engineering, the use of models is more and more recommended. A model always conforms to a unique metamodel.

Metamodeling is successfully used in building flexible modeling tools, interfaces between tools, and repository definitions.

The introduction of an intermediate metamodel can be beneficial for several reasons, namely, increased abstraction, cleaner separation between the front and back ends, and introduction of possibilities for re-targeting as well as cross-generation of code [1].

Moreover, intermediate representations may also help in supporting advanced code optimizations through ad-hoc manipulations of such artefacts achieved through opposite model transformations, which are independent of the code generating transformations.

III. DESIGNING OF ACTION LANGUAGE FOR FOUNDATIONAL UML METAMODEL

According to the abstract syntax specification given in the Alf standard [3], corresponding Ecore models were created.

In its current state, the complete Alf meta-model comprises more than 100 meta classes and thus, only some relevant cutouts can be presented here due to space restrictions.

The main abstract metaclasses related to the translation of ALF concepts are described by Statements, Expressions and Units [2].

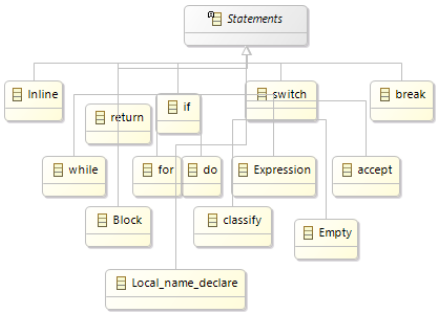

Figure 1. ALF Statements

Statements are segments of behavior that are executed for their effect and do not have values. They are the primary units of sequencing and control in the Alf representation of behavior.

This metamodel shows the statements, which are currently realized in the implementation of the Alf standard as subtypes of the abstract class Statement.

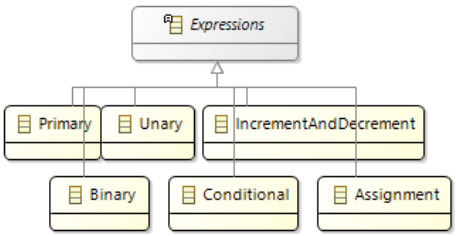

Figure 2. ALF Expressions

This Alf meta-model comprises various specializations of the meta-class Expression.

An expression is a behavioral unit that evaluates to a (possibly empty) collection of values. Expressions may also have side effects, such as changing the value of an attribute of an object.

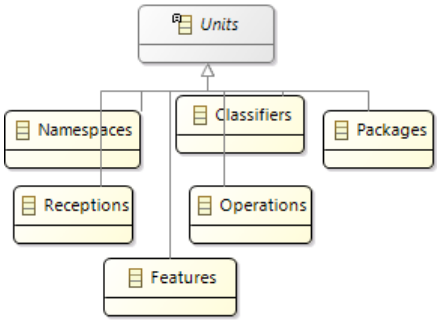

Figure 3.ALF Units

Expressions constitute the most fine-grained way of modeling in Alf and may be used in different contexts. For example, they are used for assignments, calculation, modeling of constraints or the access to operations and attributes.

Alf adds the concept of a unit to the basic UML concepts of namespaces and packages. A unit is a namespace defined using Alf notation that is not itself textually contained in any other Alf namespace definition.

Units are lexically independent (though semantically related) segments of Alf text that provide a level of granularity similar to typical programming language text files.

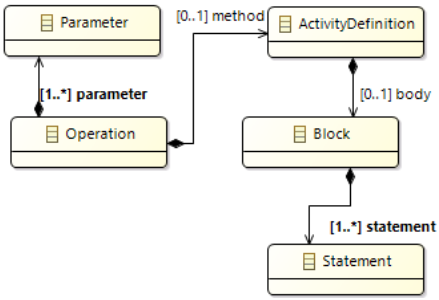

Figure 4. ALF Operations

In Alf, Operations are used for behavioral modeling. Operations may be parameterized and may contain an "Operation Method". Parameters of an operation are typed and they possess a name. Additionally, the direction of the parameter is indicated by the enumeration Parameter-DirectionKind. Possible values are in, out or inout. The type of an operation is determined by its return type. The method of an Operation contains the complete behavior realized by the operation.

One possible way of realizing this method is using a Block, which represents the body of the operation. The block itself comprises an arbitrary number of Statements.

The intermediate metamodel provides the means for hosting a wide range of concepts that are then interpreted in a certain way by the model-to-text transformation specific for the selected target programming language [1]. That is to say that, while the syntax is fixed, the semantics that the various metaconcepts assume might change from one target programming language to another and is therefore embedded in the model-to-text transformation.

IV. CONCLUSIONS

In this paper the metamodel of Action Language for Foundational UML is built with Ecore in Eclipse Modeling Framework. In this paper abstract intermediate metamodels were defined resembling common object-oriented programming languages to which ALF concepts are translated to.

This metamodels will help to understand the structure of ALF by increasing abstraction, consequently easing adaptability of transformation process.

In this paper, metamodels of Statements, Expressions, Units and Operations realized in the current implementation of the Alf standard were presented in order to highlight the properties of syntax and semantics of ALF language, and represent ALF behavioral modeling mechanism.

REFERENCES

[1] "Towards Translational Execution of Action Language for Foundational UML" – Ciccozzi F., Ciccetti A., Sjodin M., 2013
[2] "Unifying Modeling and Programming with ALF" - Thomas Buchmann, Alexander Rimer, 2016
[3] "Action Language for Foundational UML (Alf) v1.1" – Object Management Group, 2017

An Approach to Application Optimization before Compilation under Android Operation System

Volodymyr Yevtukh
Scientific adviser: PhD., Assoc. prof. Chebanyuk O.V.
Software Engineering
Institute of Computer and Informational Technologies
National Aviation University
Kyiv, Ukraine

***Abstract* - describes main concerns, methods, approaches and techniques of optimization in Unity at Android devices. The following topics will be affected: Android devices development nowadays, optimization role in software, the need of optimization, Graphics Processing Unit and Central Processing Unit part in optimization and main techniques to use.**

Keywords: optimization, Unity, Android

I. Introduction

The progress does not stand still and all branches of computer technologies are continuously evolving. Each day methods, techniques and approaches that simplify the process of development and maintenance are standing the test of time. Among with them various software appears at the market. Some of it is fated to remain unknown while other will rise and become quality standard. For latter, defining factors of its popularity are usually performance, usability, reliability and optimization. First 3 factors are certainly important but as the subject implies we will describe a particular part of the optimization which is Unity optimization at Android devices and the role it plays in development process.

II. Problem formulation

Optimization is the process of modifying a software system to make some aspect of it work more efficiently or use fewer resources [1]. It`s an important development part of any software as it enables to expand range of devices which are capable of using the software.

A. Situation nowadays

Optimization concerns arise in majority of Unity projects. Talking about Android development, the end result may vary from device to device. There are much slower and much faster phones out there, and the computational capability of mobile devices is increasing at an extraordinary rate. It is not surprising when a new generation of a mobile GPU (Graphical Processing Unit) is five times faster than its predecessor. That is incredibly fast when compared to the PC industry.

B. Mobile development

While developing a project one should remember that mobile GPUs have lots of constraints in how much heat they produce, how much power they use, and how large or noisy they can be. GPU plays great role in pixel processing and usually other powerful unit, CPU (Central Processing Unit), is unused, though many mobiles have multicore CPU. So it is often sensible to pull some work off the GPU and put it onto the CPU instead (Unity does all of these) [2].

In addition, one should remember about profiling. It can be extremely helpful in discerning which optimizations will pay off with big performance increases and which ones are a waste of your time. That means that if the CPU is slowing things down, optimizing your Shaders will not increase the frame rate at all, and if the GPU is slowing things down, optimizing physics and scripts will not help as well [2].

III. Main part

Talking about game optimization at Android, we can emphasize multiple techniques that can be used to make the game run smoother or reduce resources needed. We will start from the most obvious and simple ones and then will go to complex.

A. Scripting

1. It`s a good practice to use FixedUpdate() instead of Update() to calculate physics if you have so. The reason is FixedUpdate() is called particular amount of times per second which creates the effect of smooth position change. You can specify the time calls using Time.fixedDeltaTime variable.
2. Do not apply physics changes to colliders with no Rigidbody (Static colliders). It will cause huge leak of performance even on PC.
3. Use rigidbody.rotation instead of transform.rotation if you want to rotate your object. As it turned out it is 10-11 times faster. The same goes for rigidbody.position.
4. Try to cache links as often as possible, i.e. if you are referencing your object multiple times you can rather store a link to it in a variable and then use this variable in further scripts.
5. Avoid resource-consuming mathematical functions like Math.pow() (it seems that majority

of mathematical functions are so demanding to resources, even simple raising to power). Instead try to calculate manually.

6. Use classes for long-term objects and structures for short-term ones.

B. Unity Environment

1. Use Profiler to track which parts of your code need to be optimized. Unity Android and iOS have built in profiler. It displays messages each 30 seconds and understanding that log helps you to determine leaks in both CPU and GPU work. To open Profiler go to Window -> Profiler.
2. While building a project, use texture compression. One thing that one should remember is that different devices support different texture formats. If device doesn't support your compression format it will decompress all the textures into standard RGBA 32 and add it to already compressed textures. As a result you will lose time and resources to decompress textures but the result remain the same. One format supported by all devices that work under Android is ETC format, so you would probably like to use it in most cases.

C. Unity Scenes

1. Use simple colliders instead of mesh collider to reduce amount of vertexes required to render collider of your object.
2. Try to avoid using complex shaders. At mobile games, shaders require a lot of GPU processing so the more complex shaders you have the more time they require to be processed. Use simple shaders or write your own instead.
3. Use texture atlases and image sprites. These are collection of images/textures stored in one file. They are faster to load, have fewer state switches, and are batching friendly.
4. Bake lighting instead of using dynamic lighting. Dynamic lightning gives extra load to GPU, which may cause loss of framerate.
5. Specify GameObjects' LOD (Level of Detail) – this will make objects simpler or eliminate them completely as they move further away. At hierarchy window create empty object and add a *LOD Group* component. Now you can drag all the objects you want to fade with distance increase into this group. Each LOD group at components scale refers to visibility and mesh quality of an object. You can specify mesh for each of these LOD groups to make object look smoother when the camera is near and more rough when the camera is far.
6. Occlusion/frustum culling.

Occlusion Culling is a technique that enables Unity engine not to render objects, which are not currently seen by camera or which are overdrawn by other objects. During this process, a virtual camera will go through the scene, building a hierarchy of potentially visible objects. This information is used at the runtime by engine to define which objects should be drawn and which one should not. [3].

To set up Occlusion in your project firstly go to Window -> Occlusion Culling. Occlusion window will appear. For now, you can click on one or some objects at your scene and set them to be both Occluder and Occludee Static. This will enable them to be part of Occlusion Culling. You can also set them to be static using Inspector window.

After specifying all objects (Don't forget to Bake your Occlusion) simply start the game and go to Occlusion -> Visualization window. By rotating your camera in Game view you will see how objects appear and hide at your Scene view.

When should you use Occludee Static and when Occluder? Completely transparent or translucent objects that do not occlude, as well as small objects that are unlikely to occlude other things, should be marked as Occludees, but not Occluders. This means they will be considered in occlusion by other objects, but will not be considered as occluders themselves, which will help reduce computation [3].

Other type is frustum culling which works similar to Occlusion culling but it renders all objects in the field of view of the camera.

IV. Conclusions

Optimization can become a key feature of a game or simulation and an obstacle if you will overdo it. Before start optimizing your project, firstly think twice and ask yourself does it worthwhile. Bad optimization can cause more harm than benefit, especially if you are new at it. Nevertheless, if you decided to do it, start from general techniques and refactoring, monitor your project through profiler and define which parts of your project are resource demanding. In the skillful hands optimization become a powerful tool in development process of any game.

References

[1] Wikipedia. Program optimization. https://en.wikipedia.org/wiki/Program_optimization

[2] Unity Documentation. Optimizations. https://docs.unity3d.com/Manual/MobileOptimisation.html

[3] Unity Documentation. Occlusion Culling. https://docs.unity3d.com/Manual/OcclusionCulling.html

A Technique For Software Models Comparison

Povaliaiev D.V.
Scientific Advisor: associate professor Chebanyuk O.V.
Software Engineering Department
National Aviation University
Kyiv, Ukraine
dmytro.povaliaiev@gmail.com

***Abstract*—as software modeling gains adoption, a set of techniques to integrate it into existing software engineering practices and established lifecycles is needed. One of these techniques is a software model comparison that allows developers to easily review changes made to the software product with the help of software modeling.**

Keywords—UML, XMI, LINQ

I. Introduction

In a modern agile software development environment, time to market is both the most important and the most challenging factor for software product to achieve. This prerequisite led to recognition of importance of adaptability, which allows to decrease the time needed for product adjustments. Introduction of software modeling brings the possibility to cut the development time even further, as changes can be analyzed and verified without spending time on their implementation. Review and verification is especially important as it allows development team to identify the problems early and prevent their propagation to the code base [1][2][3]. In this context, comparison of software models facilitates reviews of changes to be made to the software.

There are several existing software tools for software models comparison, including the following ones:

- EMF Compare – provides generic comparison facilities for any kind of EMF model and allows to export differences as a model patch; the most popular one.
- SiDiff – employs similarity-based matching algorithms that allow it to be adaptable to any kind of graph-like model. Also includes a DSL to enable implementation of specialized high-level changes [1].

However, these tools require high amount of human-made operations to be conducted in order to obtain a comparison result. Such approach makes them not so suitable for automated model comparison, such as one conducted by model version control system between several change requests. Therefore, a technique suitable for automated comparison would be valuable for model versioning and change reviews.

II. Task Specification

To facilitate the process of software model change reviews and overall software model versioning, a technique for automated model comparison should be developed. The goal of this technique would be identification of the set of properties between the two models that are different. A difference is defined as an atomic add, delete, update or move operation executed on one of the model entities. Such atomic operations are collected throughout the analysis process and represented as a model themselves (a difference model) [1]. Such technique should follow the open architecture principles to make the further development of technique variations for different types of software diagrams included in models and facilitate its extension by third-party researchers and developers. In order to accomplish this task, it is necessary to do the following:

- investigate the format of software model storing;
- propose techniques for software models comparison;
- represent an algorithm for software tool working.

III. Description of developed technique

Specifically, each modeling environment tool stores and manages the models with its own internal format or its own "dialect," even if a standard format is adopted. To improve interoperability between modeling tools, standardization bodies have defined specific model interchange languages.

The best-known model interchange language is XMI (XML Metadata Interchange), a standard adopted by OMG for serializing and exchanging UML and MOF models. Unfortunately, even if this has been devised as an interchange format, different tools still adopt it with different flavors [4][5][6].

Therefore, to provide a framework for operations on XMI-compliant diagrams, the information should be processed by the Model-to-Text transformation [4][7][8][9]. In the following fragment of a class diagram representation in XMI, an example of a class diagram containing a class with an

attribute of “unlimited natural” data type and generalization connection are given:

Fragment of XMI file

```
<packagedElement xmi:type="uml:Class"
xmi:id="_qu0M8ASgEeeImrsnVV7-jw"
name="Class2">
<generalization
xmi:id="_anNPQAZYEeeIgspGntkIFA"
general="_hlUHgP8eEeaaV7q8po-h9Q"/>
<ownedAttribute
xmi:id="_Ja4tMAZYEeeIgspGntkIFA"
name="attribute1" visibility="private">
<type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPri
mitiveTypes.library.uml#UnlimitedNatural"/>
</ownedAttribute>
</packagedElement>
```

Fig. 1 Fragment of XMI file

To perform the software models comparison, they should be first transformed from the XMI-compliant text representation to the corresponding data structures via reverse Text-to-Model transformation [7][8].

After performing this processing, obtained data structures should be compared in a top-down item-by-item fashion: for each diagram in a model, its top-level entities (e.g. classes and interfaces for class diagrams) should be compared. If a top-level entity is absent in one of the models, its child entities (e.g. attributes and connections of a class on a class diagram) are marked as different ones without further investigation. After evaluating top-level entities, for each entity considered similar at this point an additional level of analysis is conducted.

There are several types of analysis that can be conducted to identify the same and different entities:

- static identity – entities are compared by explicit IDs;
- signature identity – model element features such as name, contained elements and so on are a comparison criteria;
- probabilistic function (similarity matching) [1].

To further increase the comparison effectiveness, users should be able to redefine the concept of matching for specific DSLs, so that their specific semantic can be taken into consideration.

Taking into consideration previous in the field of extraction of information from XMI-compliant software models, the .NET with built-in Language-Integrated Queries (LINQ) is proposed for implementation of this algorithm [7][8].

IV. Conclusions

A technique for software model comparison is proposed in this paper. This technique facilitates the change review process for software models that became an important artifact in agile software development process. Such improvements are expected to reduce the overall time of software product development and its time-to-market and therefor reduce the cost of development and increase project income from early launch.

References

[1] Marco Brambilla, Jordi Cabot, and Manuel Wimmer, Model-Driven Software Engineering in Practice, Second Edition. Synthesis Lectures on Software Engineering, Morgan & Claypool Publishers March 2017, Vol. 3, No. 1 , Pages 1-207

[2] Sanchez, L.E., Diaz-Pace, J.A., Zunino, A. et al. An approach based on feature models and quality criteria for adapting component-based systems Journal Software Engineering Research and Development (2015) 3: 10. doi:10.1186/s40411-015-0022-1

[3] Sielis, G.A., Tzanavari, A. & Papadopoulos, G.A. ArchReco: a software tool to assist software design based on context aware recommendations of design patterns Journal Software Engineering Research and Development (2015) 2. doi:10.1186/s40411-017-0036-

[4] XML Metadata Interchange access mode http://www.omg.org/spec/XMI/

[5] Architecture-Driven Modernization™ (ADM™): Abstract Syntax Tree Metamodel™ (ASTM™) http://www.omg.org/spec/ASTM/1.0

[6] Philip Newcomb. Chief Executive Officer Abstract Syntax Tree Metamodel Standard ASTM Tutorial 1.0 access mode: http://www.omg.org/news/meetings/workshops/ADM_2005_Proceedings_FINAL/T-3_Newcomb.pdf

[7] Chebanyuk E. V., Povaliaiev D. V., Software architecture verification approach, Software Engineering, National Aviation University, №2 - 2017

[8] Elena Chebanyuk and Kyryl Shestakov An Approach for Design of Architectural Solutions Based on Software Model-To-Model Transformation International Journal "Information Theories and Applications", ISSN 1310-0513 Vol. 24, Number 1, 2017, pages 60-84

[9] Chebanyuk E. V., Povaliaiev D. V., An approach for architectural solutions estimation, "International journal Informational theories and knowledge", Vol 11, number 2– 2017.

Аналіз сучасного стану досліджень в галузі штучного інтелекту в Україні

Шевчук О.О.
науковий керівник: Конрад Т. І.
Кафедра інженерії програмного забезпечення
Навчально-науковий інститут комп'ютерних інформаційних технологій
Національного авіаційного університету
Київ, Україна
alexeyolegovichsh@gmail.com

***Анотація* — робота присвячена аналізу стану досліджень в галузі штучного інтелекту в Україні. Наведено інформацію про досягнення, розробки та винаходи вітчизняних вчених, та основні напрямки розвитку штучного інтелекту в Україні.**

Ключові слова — штучний інтелект, розпізнавання образів та мови, дослідження, розвиток, технології розпізнавання, пошуовий сервіс

I. Вступ

Штучний інтелект (ШІ) – один з найновіших напрямів інформаційних технологій (ІТ). Передісторія створення людиноподібних механізмів починається ще в стародавньому світі і проходить складний шлях еволюції: від мрій і легенд, перших андроїдів, механічних шахових гравців та інших складних механізмів з людською поведінкою до сучасних інтелектуально-механічних роботів. Перші дослідження, що відносяться до ШІ у сучасному розумінні були зроблені майже відразу ж після появи перших обчислювальних машин [1].

II. Постановка проблеми

Все частіше надходять новини про досягнення у сфері використання штучного інтелекту в самих різних галузях життєдіяльності людини. Таксі зі штучним інтелектом в Японії [2], штучний інтелект на фондових біржах в США, в медицині китайської компанії Watsons [3] – і це далеко не весь список розробок. Тому, актуальним питанням є аналіз наукових досягнень в галузі штучного інтелекту в Україні.

III. Основна частина

Великого успіху в сфері розпізнавання образів досягнула компанія Viewdle. Фахівці компанії займались розробкою технологій розпізнавання осіб і об'єктів, які широко використовуються у пошукових сервісах. Результатами наших співвітчизників зацікавилася корпорація Google. Співробітники Viewdle переїхали до США, де увійшли в команду Motorola Mobility і Motorola ATAP group [4].

Також стартап Looksery досягнув успіху в цій сфері. Looksery - це програма для мобільних пристроїв, яка дозволяє в реальному часі змінювати власне обличчя так, щоб красиво виглядати на фото і навіть під час відеочату. Окремий додаток Looksery зник з AppStore, ставши частиною програми Snapchat [5].

У 2009 році українці Олексій Шевченко, Максим Литвин та Дмитро Лідер заснували компанію Grammarly, що займалася розробки однойменного сервісу для перевірки англійського правопису. Станом на 2017 рік Grammar займав перше місце як «найкращий онлайн-сервіс з перевірки правопису» від TopTenReviews, з рейтингом 9.68. Станом на 2017 рік команда Grammarly становить понад 100 осіб, а кількість щоденних активних користувачів досягає майже 7 мільйонів. Сервіс підвищує якість письмового спілкування, пропонуючи виправлення, які роблять текст граматично, стилістично і структурно правильним [6].

IV. Висновки

В роботі проаналізовано стан та основні напрацювання українських вчених в галузі штучного інтелекту. Визначено основні напрямки та перспективи досліджень з подальшого впровадження ШІ в наукових цілях та для реалізації прикладних задач.

Список використаних джерел

[1] Подгаєцький О. О. Еволюція розробок у галузі штучного інтелекту в Україні та світі/ О. О. Подгаєцький. // Дослідження з історії техніки. – 2012. – №16. – С. 48–54. – [Електронний ресурс] – Режим доступу: http://ela.kpi.ua/bitstream/123456789/7703/1/RHT-issue-16-title-05-Podgayetsky.pdf

[2] Никитин А. Sony создала компанию для внедрения ИИ на рынок такси [Електронний ресурс] / Артем Никитин // АНО "Иннополис Медиа". – 2018. – Режим доступу: https://hightech.fm/2018/02/21/sony-ai-taxi.

[3] Delo.ua [Електронний ресурс]: [Інтернет-портал]. – Електронні дані. – Режим доступу: https://delo.ua/lifestyle/kakuju-kareru-mozhet-sdelat-iskusstvennyj-intellekt-333683/. – Назва з екрана.

[4] Viewdle [Електронний ресурс] // Вікіпедія: вільна енцикл. – Електрон. дані. – Режим доступу : https://uk.wikipedia.org/wiki/Viewdle. – Назва з екрана. – Дата останньої правки: 18.10.2015.

[5] Delo.ua [Електронний ресурс]: [Інтернет-портал]. – Електронні дані. – Режим доступу: https://delo.ua/business/snapchat-kupil-odesskij-startap-za-150-millionov-303880/. – Назва з екрана.

[6] Grammarly [Електронний ресурс] // Вікіпедія: вільна енцикл. – Електрон. дані. – Режим доступу :

https://uk.wikipedia.org/wiki/Grammarly. – Назва з екрана. – Дата останньої правки : 07.11.2017.

Дослідження основних підходів покращення розробки архітектури програмного продукту

Котенко Денис Валентинович
науковий керівник: Конрад Тетяна Ігорівна
Кафедра інженерії програмного забезпечення,
Навчально-науковий інститут комп'ютерних інформаційних технологій,
Національний авіаційний інститут
Київ, Україна
d.kotenko@elgnc.com

***Анотація* — робота присвячена дослідженню процесу проектування та розробки архітектури програмного продукту, з метою покращення якості розробки на етапі проектування та конструювання. Запропоновано методики та підходи для підвищення ефективності організації проектування, розробки архітектури та підтримки програмного продукту.**

Ключові слова — програмне забезпечення, архітектура програмного забезпечення, підходи розробки архітектури, розробка програмного забезпечення, ефективність системи, гнучкість системи, маштабованість системи, підтримка програмного продукту, рефакторинг.

I. Вступ

IT індустрія бурхливо розвивається, що призводить до постійного зростання кількості програмних продуктів (ПП). Щорічно створюється та збільшується обсяг новітніх систем для потреб бізнесу та державних установ.

Крім створення нових ПП важливою складовою IT ринку є підтримання вже розроблених продуктів, розширення їхньої функціональності та адаптації до сучасних вимог. Все це призводить не тільки до збільшення розмірів таких систем, а й до ускладнення їхньої структури та труднощів в подальшій підтримці. Нерідко доводиться підтримувати розроблену систему протягом багатьох років, або навіть десятиліть.

Підтримка складних систем є непростою задачею і потребує значно більших трудових та матеріальних ресурсів ніж зазвичай планується. Розроблення нових функцій та доопрацювання створеної функціональності постійно супроводжують розробників на етапі підтримки. Відповідно, ціна навіть незначних змін може виявитись суттєвою, а внесені зміни вплинути на роботу всієї системи. Тому для компаній з розробки програмного забезпечення питання про зменшення кількості ресурсів необхідних для розробки продукту та підвищення його ефективності є вкрай актуальним, особливо в умовах обмеженого бюджету.

II. Постановка проблеми

Поставлені вимоги перед розроблюваною системою нерідко змінюються. Внесення змін можливе як на етапі проектування та розробки, так і після вводу в експлуатацію. Розроблені модулі системи після завершення розробки можуть виявитись неактуальними по відношенню до бізнес потреб. В таких випадках систему відправляють на доопрацювання для внесення змін у вже в розроблений продукт. Доопрацювання системи на такому етапі стає більш складним та затратним процесом.

Великою проблемою для компаній розробників є суттєва втрата якості ПП за умови суттєвих змін вимог замовника, що не можуть бути задоволені в рамках архітектурних рішень. Експерт з програмних рішень Г. Буч порівнює внесення суттєвих змін у вже побудовану архітектура з добудуванням двох поверхів у вже збудованому хмарочосі і, причому не зверху, а збоку [1].

Для попередження незапланованих витрат на внесення змін в розроблений та введений в експлуатацію ПП доцільним є ефективна організація процесу проектування архітектури на етапі планування, та дотримання всією командою розробки певних правил, що підвищить кінцеву якість продукту.

В таких випадках правильно вибрані архітектурні рішення на початку проектування системи в майбутньому зможуть зменшити час та обсяг робіт по внесенню нових змін, розширення функціональності та виправленні помилок. Таким чином це дасть можливість зменшити обсяг капіталовкладень в проект.

III. Основна частина

Основним завданням архітектури є зробити процес розробки та супроводження більш простим та ефективним. Систему в якій продумана архітектура простіше тестувати, вносити зміни та відлагоджувати.

Обрані архітектурні рішення є ключовими факторами, що впливають на загальну якість продуку. Під якістю ПП розуміється відповідність вимогам та потребам замовника.

На прикладі успішних проектів можна виділити декілька основних, універсальних підходів, які характеризують добре продуману архітектуру:

A. Ефективність системи

Важливою особливістю є виконання поставлених завдань системою, при чому в різних умовах. До цього пункту відноситься надійність, продуктивність, безпека та маштабованість.

B. Гнучкість системи

При розробці будь-якої системи бувають етапи, коли вимоги змінюються чи додаються нові. Відповідно чим швидше можна внести зміни в існуючий функціонал і чим менше помилок при цьому виникне – тим більш гнучкою буде система. Тому при розробці важливо оцінювати на скільки багато в подальшому може бути змін та якого роду. Важливо себе запитати: «Що, якщо обраний підхід виявиться невірним і як багато прийдеть переписувати?». Також важливою концепцією є те, щоб зміна однієї частини системи не призводила до змін інших та не впливала на їх працездатність.

Роберт Мартін на протязі багатьох років аналізу ефективності розробки ПП дійшов висновку, що варто притримуватись гнучких методологій розробки при конструюванні, що дозволить в короткі терміни постачати ПП з заявленою функціональністю [2]. Розробка згідно з таким підходом повинна відбуватись короткими ітераціями, та процес розробки буде гнучко адаптуватись під нові вимоги [3].

C. Можливість розширення

На початковому етапі в систему варто закладати тільки онсовний функціонал, припускаючи що додаткова інфраструктура взагалі не знадобиться. Додавати її по мірі необхідності, або якщо є докази на користь того, що включити нову інфраструктуру обійдеться легше та дешевше, ніж відкласти на майбутнє [2]. При цьому архітектура повинна легко нарощувати новий функціонал по мірі необхідності.

Властивості гнучкості та розширюваності архітектури на стільки важливі, що винесені в окремий принцип – «Принцип відкритості/закритості»: програмні сутності повинні бути відкриті для розширювання та закритими для модифікації.

D. Маштабування процесу розробки

Архітектура повинна розподілити процес розробки таким чином, щоб за рахунок нових людей можливо було скоротити терміни постачання продукту.

E. Тестованість

Код покритий тестами буде мати менше помилок та буде надійніше працювати. Була створена ціла методологія розробки через тестування – Test driven development (TDD) [4].

F. Можливість повторного використання

Варто систему розробляти таким чином, щоб її можна було використати в інших проектах з схожою предметною областю.

G. Підтримка та добре структурований, зрозумілий програмний код

Часто трапляється так, що під час розробки одні працівники замінюються іншими, а вести підтримку системи доводиться новим працівникам, що не брали участі в її початковій розробці. Тому архітектура повинна надати можливість швидко та легко вникати в проект новим розробникам. Проект не повинен мати дубльованого коду, натомість бути добре структурованим і мати добре описану документацію. Варто зауважити, що велика кількість документації на стільки ж погана як і повна її відсутність. Крім того документація повинна бути завжди актуальною і постійно оновлюватись [2].

По мірі того як додаються нові функції і виправляються помилки, структура коду погіршується. Якщо не звертати на це увагу, то така деградація призведе до плутанини, яку неможливо підтримувати. Такий код ще часто називають спагеті код (заплутаний код).

В цілях підвищення якості програмного коду рефакторинг має проводитьсь постійно, кожну годину, а то й навіть кожні півгодини. Використовуючи рефакторинг програмний код стане елегантним, виразним та простим в розумінні.

Крім того варто дотримуватись єдиного стандарту кодування всією командою. Весь код повинен виглядати так, ніби його писала одна – дуже кваліфікована людина.

Важливо також пам’ятати, що простота – це майстерність досягати більшого, роблячи менше. А робоча система – основний показник успішності проекту [2].

IV. Висновки

В статті обгрунтовано важливість етапу проектування архітектури при розробці програмного забезпечення. Також виділено основні підходи покращення розробки архітектури ПП, такі як: ефективність, гнучкість, розширюваність, маштабованість, тестованість, можливість повторного використання та супроводжуваність системи, дотримання яких дозволять суттєво покращити процес розробки та підтримки ПП. Описані підходи можуть застосовуватись для всіх мов програмування і повинні дотримуватися протягом розробки всього проекту. Дотримання даних принципів суттєво покращить якість роботи продукту, зменшить час розробки і, відповідно, знизить кінцеву вартість.

Список використаних джерел

[1] Буч Г. Объектно-ориентированное проектирование с примерами применения – К.: Диалектика; М.: Конкорд, 1992. — 519 с.

[2] Мартин Р., Мартин М. Принципы, паттерны и метрики гибкой разработки на языке C#. – Пер с. англ. – СПб: Символ-плюс, 2011. – 768с.

[3] Эндрю С., Дженнифер Г. Постигая Agile. Ценности, принципы, методологии. – Пер с. англ. – СПб: Манн, Иванов и Фербер, 2017. – 573с.

[4] James Bender, Jeff McWhether Professional Test Driven Development with C#. – Wiley Publishing, Inc. – 327p.

Investigation of Unity3D timers performance

Krainy Mykyta
ICIT SE-318
Scientific adviser: PhD., Assoc. prof. Chebanyuk O.V.
Software Engineering Department
National Aviation University
Kyiv, Ukraine
nickkhryne@gmail.com

***Abstract* — investigation of effectiveness and performance of different ways to organize timers in Unity3D is represented in this paper. The grounding of choosing approach, using Time.deltaTime to estimate time spent in different game processes is given.**

Keywords—Unity3D, timer, coroutine, InvokeRepeating, Time.deltaTime.

I. Introduction

Why do we need time calculation in games?

Dealing with different applications and games, it is often needed to perform time calculation. It can be used to resolve gaming issues, such as start and finish of game, to calculate how much time is wasted on some actions in the game or to define the place of user in highscore table. In different applications timers can be used for various purposes, beginning from the starting stopwatch to the simple calculation of passed time.

The most important problem in Unity3D is to perform precise time calculation which will work with stability. The different approaches to resolve this problem are highlighted in this investigation.

II. Aim of research

The aim of research is to investigate ways of organizing time interval counting in Unity3D. After that provide comparative analysis and define what approach to calculate time allows obtaining precise results.

III. pros and cons review of different means to organize time calculation in unity3d

A. *Coroutines*

Coroutines are a general control structure whereby flow control is cooperatively passed between two different routines without returning. When we call a function, it runs to completion before returning. This effectively means that any action taking place in a function must happen within a single frame update; a function call can't be used to contain a procedural animation or a sequence of events over time. The execution of a coroutine can be paused at any point using the yield statement. The yield return value specifies when the coroutine is resumed. Coroutines are excellent when modelling behaviour over several frames. Coroutines have virtually no performance overhead. StartCoroutine function always returns immediately, however you can yield the result. This will wait until the coroutine has finished execution. There is no guarantee that coroutines end in the same order that they were started, even if they finish in the same frame. Example of applying coroutine to calculate 5 seconds is given below

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour
{
    IEnumerator WaitAndPrint()
    {
        // suspend execution for 5 seconds
        yield return new WaitForSeconds(5);
        print("WaitAndPrint " + Time.time);
    }

    IEnumerator Start()
    {
        print("Starting " + Time.time);

        // Start function WaitAndPrint as a coroutine
        yield return StartCoroutine("WaitAndPrint");
        print("Done " + Time.time);
    }
}
```

B. *InvokeRepeating*

Invokes the method in `time` seconds, then repeatedly invokes it every repeatRate seconds. CancelInvoke method stops the repetition. The method in invoked only at regular intervals. Using irregular intervals, multi-step is impossible in this kind of time evaluation. Also, despite of its simplicity, it

can have huge influence on performance, compared to the Coroutine method. Invoke* functions in Unity3D are implemented by means of reflection, which has huge overhead when calling, and it should be avoided. Example of applying Invokerepeating is given below

```
using UnityEngine;
using System.Collections.Generic;

// Starting in 2 seconds.
// a projectile will be launched every 0.3 seconds

public class ExampleScript : MonoBehaviour
{
    public Rigidbody projectile;

    void Start()
    {
        InvokeRepeating("LaunchProjectile", 2.0f, 0.3f);
    }

    void LaunchProjectile()
    {
        Rigidbody instance = Instantiate(projectile);

        instance.velocity = Random.insideUnitSphere * 5;
    }
}
```

C. *Time.deltaTime*

The time in seconds it took to complete the last frame It is used to make game frame rate independent. If a value is added or subtracted value every frame chances are we should multiply with Time.deltaTime. When we multiply with Time.deltaTime we essentially express: “I want to move this object 10 meters per second instead of 10 meters per frame”. Time.deltaTime should not be relied on from inside OnGUI since OnGUI can be called multiple times per frame and deltaTime would hold the same value each call, until next frame where it would be updated again.

IV. Grounding of choosing approach for precise time calculation in unity3d

When different techniques of timer implementation in Unity3D are considered, the most stable and productive mechanism should be preferred. As it was stated above, InvokeRepeating has huge overhead, so in large projects it influences the performance too much.

In case of coroutines, the logic doesn’t block the main thread, the performance isn’t reduced too much. Unlike threads read/write is in series rather than concurrent parallel reads, which means that using coroutines is safer. However, it can cause stack memory issues if a coroutine is killed externally or never fully executed.

If we can neglect some time deviations, the coroutine approach is the most obvious to use. However, if precise calculation is needed, time.deltaTime approach should be considered. Using time.deltaTime in project, it is made frame independent, so performance issues should not affect heavily time calculation. In Unity3D, a simple time calculation in Update() method can look as in example:

```
public Text timerText;
private float secondsCount;

void Update()
{
    UpdateTimerUI();
}

public void UpdateTimerUI()
{
    secondsCount += Time.deltaTime;
    timerText.text = (int)secondsCount + "s";
}
```

It is an example of simple timer, measuring seconds. The main advantages of time.deltaTime approach are precision, simplicity and absence of additional load on performance. So, in usual cases of time calculation, it is recommended to use this approach.

Conclusion

The aim of investigation is to define the most exact way to estimate time in Unity3D applications. It was defined that that time.DeltaTome approach allows to obtain the most precise results. But it can be applied only for calculation of very small amount of time. To calculate more large intervals of time it is recommended to use technique that is based on “invokeRepating” implementation. Courotines usually give errors when application works in background. Also In order to check results of calculating time involving “InvokeRepeating” technique you may use StopWatch class.

References

[1] “Unity in action” Joseph Hocking

[2] https://docs.unity3d.com/ScriptReference/MonoBehaviour.InvokeRepeating.html

[3] https://docs.unity3d.com/Manual/Coroutines.html

[4] https://docs.unity3d.com/ScriptReference/Time-deltaTime.html

Research and Development Workstation Environment: the new class of CRIS

Kyrylo Malakhov
Scientific advisor: Vitalii Velychko
Microprocessor Technology Department,
V.M. Glushkov Institute of Cybernetics,
The National Academy of Science of Ukraine,
Kyiv, Ukraine
malakhovks@nas.gov.ua

***Abstract* — against the backdrop of the development of modern technologies in the field of scientific research, the new class of Current Research Information Systems and related intelligent information technologies have arisen. It was called – Research and Development Workstation Environment – the comprehensive problem-oriented information systems for scientific research and development lifecycle support. The given paper describes general information model of the Research and Development Workstation Environment class systems.**

Keywords — CRIS, RDWE, Research and Development, Composite Web Service, Cloud Computing.

I. Introduction

The development of modern technologies increasingly covers the field of intellectual activity and, especially, in the field of scientific research and development. The new class of *Current Research Information Systems* (CRIS) and related intelligent information technologies have arisen that support the main stages of the scientific research and development lifecycle, starting with the semantic analysis of the information & data material of arbitrary domain area and ending with the formation of constructive features of innovative proposals. It was called – *Research and Development Workstation Environment* (RDWE) – the comprehensive problem-oriented information systems for scientific research and development support. A distinctive feature of such systems and technologies is the possibility of their problematic orientation to various types of scientific research and development by combining on a variety of functional services and adding new ones within the Cloud-integrated Environment (inside the Ubuntu open source operating system as the integrating cloud environment for instance). The given paper describes general information model of the RDWE class systems.

II. Current Research Information Systems

In the modern English-speaking scientific environment, a steady term – Current Research Information System (CRIS) [1] was introduced to designate scientific information systems for access to scientific and academic information. It is important to emphasise that the definition of CRIS also specifies that CRIS is not only intended for direct access to information sources of science but also, according to the ERGO project, for:

- to facilitate access to national scientific and technical information services;
- to identify the main existing information sources and to evaluate access possibilities and the potential for the utilization of these sources at European level;
- to invite national data hosts to offer their Research and Development (R&D) information and to make this information searchable for the user.

The EuroCRIS organisation was founded in 2002, is an international not-for-profit association, that brings together experts on research information in general and research information systems. The mission of EuroCRIS is to promote cooperation within and share knowledge among the research information community and interoperability of research information through CERIF – the Common European Research Information Format. Areas of interest also cover research databases, CRIS related data like scientific datasets, (open access) institutional repositories, as well as data access and exchange mechanisms, standards and guidelines and best practice for CRIS. The EuroCRIS provides the framework for the flow of information/data between a broad variety of stakeholders: researchers, research managers and administrators, research councils, research funders, entrepreneurs, and technology transfer organisations.

Requirements for CRIS in the information management aspect of strategic research management has been analysed in [2], which describes the types of research managerial activities, introduces the currently available information sources and how the information found in these is applied today. The paper [3] describes requirements for CRIS to effective dissemination of technologies, the management of scientific programs and functioning of funds for research funding. Noted the importance of CRIS systems to collaborate researchers and to support the information functioning of funds. Also represented the basic lifecycle of scientific programs and the information demands of participants at each phase of the cycle.

The CORDIS portal emphasised the basic use cases of scientific portals for researchers: keep up-to-date on current

research findings and strategic directions; identify funding sources for R&D; find partners to cooperate in R&D activities and share expertise; form transnational consortia for exploitation of research results; promote and locate transferable technologies, and more.

III. General information model o the RDWE class systems

The RDWE class system's generalized information model represented as a three-tuple composite web service (CWS) using the revised formalism given in [4]:

$$CWS = \langle AWS, F, CI_{Env} \rangle,$$

where:

is the RDWE composite web service;

$AWS = \{aw$ is a set of atomic web services (problem-oriented microservices and FLOSS applications; personalized FLOSS applications) available for usage. The set consists of the problem-oriented atomic web services and each of them can be designed and developed as a microservice or a desktop application, that allows them to be used as an independent software separately from the RDWE and as its components inside ;

$F = AWS$ is a set of functions, the functional filling-up of the RDWE, each function is the result of coordination and interaction of the elements.

$C_j \subseteq AWS, C_j = \{aws_k | k \geq 1, k$ is a subset of atomic web services that are required to implement the -th function of ;

$CI_{Env} = \{prl, mid, os, crd, typ$ is a set of elements (represented as layers) that combine into the Cloud-integrated Environment (CIE);

– *physical resource layer* represents physical hardware and facility resources;

– *middle layer* (using in the concepts of cloud service orchestration model [5]) represents resource abstraction and control layer. It is supposed to use OpenStack software platform;

– *operating system layer* represents guest operating system. It is supposed to use Ubuntu server with LXDE (abbreviation for Lightweight X11 Desktop Environment) desktop environment or Xfce desktop environment. Atomic web services work on the operating system layer – ;

– *coordination component*. The function is to coordinate atomic web services in , and by the coordination procedure we will understand the execution of invocation of some in the defined sequence. The coordination component can be implemented as the reverse proxy server of tasks. Nginx also is a part of – used as front-end to control and protect access to the server on a private network, performs tasks such as load-balancing, authentication, decryption, and caching.

– *a typical FLOSS layer* includes some regular application suit needed for the scientific research and development lifecycle (regular software suit may change in the future): LibreOffice office suite; Mozilla Firefox and Chromium web browsers; Sylpheed email client; Sublime Text 2 and jEdit source code editors; Wine compatibility layer that aims to allow computer programs developed for Microsoft Windows to run on Unix-like operating systems; Python (SciPy Python library used for scientific computing and technical computing); R environment for statistical computing and graphics; Eclipse integrated development environment; Redmine project management and issue tracking tool; X2Go remote desktop software.

CIE of RDWE delivers to researchers (to researcher's client device – laptop, desktop, mobile or tablet) using the extended *Platform-as-a-Service* service delivery model via X2Go remote desktop software and ssh cryptographic network protocol (Fig. 1).

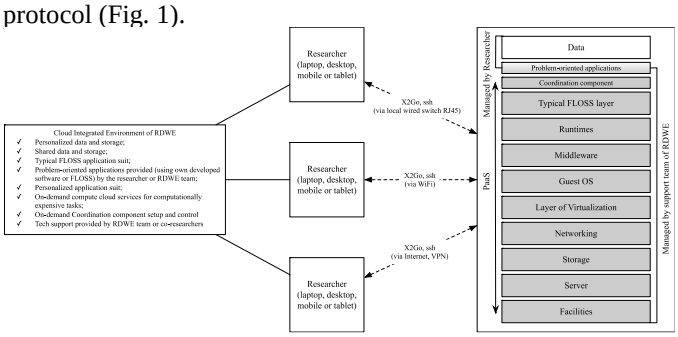

Fig. 1. Cloud-integrated Environment of RDWE delivery model.

To take all features of CIE, the researcher's client device (laptop or desktop) must run latest stable release of X2Go remote desktop software and comply with the following system requirements.

IV. Conclusion

The development of modern technologies increasingly covers the field of intellectual activity and, especially, in the field of scientific research and development. The existing CRIS oriented on the following main types of services: access and reuse of scientific and academic information, methodologies, and technologies; information search; targeted dissemination of information; messaging services; bridging of horizontal and vertical relations between scientists; backup data storage and archival information. We propose the new class of CRIS and related intelligent information technologies. This class supports the main stages of the scientific research and development lifecycle, starting with the semantic analysis of the information of arbitrary domain area and ending with the formation of constructive features of innovative proposals. It was called – Research and Development Workstation Environment – the comprehensive problem-oriented information systems for scientific research and development support. A distinctive feature of such systems is the possibility of their problematic orientation to various types of scientific activities by combining on a variety of functional services and adding new ones within the Cloud-integrated Environment.

References

[1] Nikos Houssos. Cris for research information management. 2011.

[2] Niclas Lindgren and A Rautamki. Managing strategic aspects of research. Proceedings CRIS-2000, Helsinki, 2000.

[3] Peter Dew, Christine Leigh, and Bill Whyte. Adviser ii:theory and practice of finding and presenting rtd results. Proceedings CRIS-2000, 2000.

[4] V. Shkarupylo, R. Kudermetov, and T. Paromova. Conceptual model of automated composite web services synthesis process. 2012.

[5] Sandeep Bhowmik. Cloud Computing. Cambridge University Press, 2017.

A technique for software models merging

Mironov Yuriy
Scientific adviser: PhD., Assoc. prof. Chebanyuk O.V.
Software Engineering Department
Institute of Computer Informational Technologies
National Aviation University
Kiev, Ukraine
Yuriymironov96@gmail.com

***Abstract* — in modern software engineering industry the technologies change and evolve rapidly. Because of this, nowadays software architects experience a strong need to express software solution not as a piece of code, but as a software model that can be described, extended and re-worked with no affiliation to programming language. Therefore, there is strong need in tools and comprehensive environment for working with software models. The paper considers a problem of software models merging and introduces an algorithm for UML diagram processing.**

Keywords — UML. LINQ, XMI, XML, Abstracr Syntax Tree

I. Introduction

Software engineering is a competitive and quickly developing industry that is prone to changes. However, the majority of these changes involve only the technology stack used to build a solution. At the same time, the principles and approaches to software design, as well as fundamental rules of how digital solutions work, remain the same, despite the programming language used for implementation change year by year. In such dynamic environment, software models represent the only point at which a product can be clearly seen: it has just right amount of technical details to understand all the peculiarities but at the same time it is not attached to any platform or programming language. So, while technologies deprecate, the software models live on and can be implemented with minimum effort, because all the architecture is ready and described in comprehensive form.

Another peculiarity of modern software engineering is a wide spread of Agile development methodology, that implies readiness to frequent change of requirements to a distinct solution. This often brings turmoil to the project, prevents from seeing the bigger picture and making right decisions in terms of software architecture. Software models are useful in this case, because they are more human-readable, understandable and easy to edit immediately seeing how changes affect the entire solution.

Therefore, modern driven development is heavily used in modern software engineering and software models may be considered to be the key artifacts of software development process. Being so important for the project success, software models should be supplemented with a comprehensive work environment, like IBM Rational Software Architect (IBM RSA). However, for more comfortable collaborative work with software models, there is a strong need for a tool for merging software diagrams. Software models merging is a major concern and is hard to implement properly, considering all the pitfalls. A given paper intends to give a short overview of software models merging problem, derive a basic approach and algorithm that can grant a merging software models.

II. Formulation of the problem

The main task is to design an approach to software models merging. The “merge” process implies input of several diagrams and a single diagram as an output. The input diagrams should be of the same type and should have some similar parts for successful merge. The main goal of such an approach is to merge software models identifying duplicates and elements with strong similarities. Duplicates must be unified into a single element.

Since the software models are divided into several types by their concept, distinct logic for comparison and merge should be described for different diagram types (e.g. activity diagrams, flow diagrams et cetera). However, the common parts of merging technique for different software models processing should be discovered and unified to design more comprehensive solution.

III. Main part

In order to read software models programmatically, one should discover a file/markup format in which the model is stored. There are many applications for work with Unified Modeling Language (UML) diagrams and various respective storage formats (StarUML and JSON, UMLet and custom eXtensible Markup Language dialect et cetera). The most notorious of these is IBM RSA application. It introduces a comprehensive environment for work with multiple diagram types with advanced features like metamodeling tools and model execution. It uses XML Metadata Interchange (XMI) to store UML diagrams. XMI is an open standard developed by Object Management Group used for storing complex consistent data. In the paper, XMI storage (v2.x) format is considered for software models processing [2].

The XMI is expressive enough to be able to describe various types of UML diagrams. In scope of the paper,

package diagram will be considered. In any case, XMI describes diagram as a set of entities and their relations. Entities are stored using fundamental <packagedElement> tag [1]. PackagedElement may be class or interface in class diagram, actor or use case in use case diagram, code package in package diagram or component in component diagram. Also, every packagedElement has xmi:id attribute that stores a unique identifier of each entity. This should be kept in mind because it may be used in case of collaborative editing of existing entity of a diagram (such identifier may be used as pivots for merging).

Moreover, packagedElement is used to store relations between these elements, as well as entities, have own identifiers and attributes that show their types, and also they have two nested tags that point out to packagedElements by storing their identifiers. This allows to depict entities and their relations [3].

Example of package description in package diagram:

<packagedElement xmi:type="uml:Package" xmi:id="PackageId" name="PackageName"/>

And here is an example of connections between packages:

<packagedElement xmi:type="uml:Use" xmi:id="SomeUseId" memberEnd="UsedPackageEndId UsingPackageEndId">

<ownedEnd xmi:id="UsedPackageEndId" type="UsedPackageId" />

<ownedEnd xmi:id="UsingPackageEndId" type="UsingPackageId" />

</packagedElement>

So, it is possible to parse XMI that stores model data and recognize a set of entities represented on the diagram. Having entities (and their identifiers respectively) in memory, it is possible to recognize and remember relations between models. This allows to handle the case when there is a need to merge diagrams where new relations have been defined: entities with similar identifiers exist on both diagrams and it is not a problem. The identifiers themselves allow to match entities from different diagrams and merge non-conflicting attributes (like new-added package description).

Moreover, because of the generic approach of XMI to storing entities and their relations, there is no need to design different approaches to different UML diagram types, because programmatically they are the same and can be recognized via similar parsing techniques.

However, for now there is no way to introduce smart merges like checking how to merge changes in a single field of a single model. However, this is not possible even in mature version control systems like Git. Still, it is a good idea to mark it as a merge conflict, just as Git does.

As for implementation, .NET Framework and its LINQ library can be used for fulfilling such task. LINQ is a powerful tool for parsing XML and it suits well for parsing XMI [4]. Having XMI tags and attributes extracted in memory, it is not hard to arrange data into domain-specific classes and traverse them, conducting the merge algorithm.

Considering all the aforementioned, here is a description of merge algorithm itself:

- Traverse input XMI files, recognize all the packageElements that represent diagram entities (classes, packages et cetera).
- Put the entites in two key-value storage for further mapping, where key is XMI unique identifier.
- Traverse both XMI files recognizing packageElements representing relations, put them into distinct key-value storages.
- Go through key-value storages with entities, find matching identifiers, create new resulting entity for each match in both storages that represents a union of attributes of both input entities.
- Go through storages with relations, apply missing relations from both of input model into output model.
- Serialize the in-memory representation of entities and their relations in the XMI format (output file).

This is only a skeleton of algorithm that handles only the most simple and naive cases. However, it is still a solid fundament for more logic that can improve model merge flow. Additional flags, parameters and options may help to automatically resolve model merge conflict.

IV. Conclusions

Using XMI as a machine friendly representation of UML diagrams, it appeared to be simple to recognize entities and their relations. Moreover, XMI introduced software models in generic format, which allows to derive a general approach to software models merging. The algorithm introduced in the paper is simple to implement, basic yet comprehensive and covering the majority of models merge use cases.

References

[1] Object Management Group, XML Metadata Interchange Specification, http://www.omg.org/spec/XMI/2.5.1/, 2015

[2] Marco Brambilla, Jordi Cabot, and Manuel Wimmer, Model-Driven Software Engineering in Practice, Second Edition. Synthesis Lectures on Software Engineering, Morgan & Claypool Publishers March 2017, Vol. 3, No. 1, Pages 1-207

[3] E. Chebanyuk, Yu. Mironov. An approach of obtaining initial information for software models analysis. “International journal. Informational content and processing.”, Vol. 4, number 2 – 2017

[4] E. Chebanyuk, Yu. Mironov Extracting information about software models. Інженерія програмного забезпечення. №1 -2017

Кросплатформеність. Огляд платформи .NET Core

Мороз О.Д.
науковий керівник: Скалова В.А
Кафедра інженерії програмного забезпечення,
Інститут комп'ютерних інформаційних технологій
Національний авіаційний університет,
Київ, Україна
elenad.moroz@gmail.com

***Анотація* — робота присвячена розгляду проблеми кросплатформної розробки засобами .NET. Ця стаття повинна прояснити, що таке .NET Core сьогодні, які цілі нової реалізації .NET та як вона пов'язана з Microsoft .NET Framework. У роботі досліджені основні можливості, характеристики та переваги .NET Core. Також в роботі розглянуті сценарії платформи та набір інструментів для розробки відповідних застосунків.**

Ключові слова — .NET Core, CoreCLR, кросплатформеність, програмне забезпечення, Windows, Mac OS, Linux, IDE, Microsoft Visual Studio, Visual Studio Code, Visual Studio for Mac, JetBrains Rider.

I. Вступ

Кросплатформеність – здатність програмного забезпечення працювати більш ніж на одній платформі або операційній системі [1].

Основним завданням IT-компаній є забезпечення їхнього успіху на ринку програмного забезпечення. Кросплатформна розробка програмного забезпечення допоможе вирішити багато важливих моментів для подальшого успіху продукту. Наприклад, кількість кінцевих користувачів, простота у встановленні та освоєнні програми.

Одним з останніх прикладів кросплатформних систем є новий реліз .NET Core. З'явившись в 2002 році, платформа .NET пройшла довгий шлях і змогла завоювати сильні позиції як надійне рішення корпоративного рівня [2].

27 червня 2016 року [4] разом з Microsoft Visual Studio 2015 Update 3 був випущений .NET Core версії 1.0. В оновленні для VS 2015 була реалізована підтримка розробки під .NET Core.

.NET Core є кросплатформеним (Windows, Mac, Linux) аналогом .NET Framework з відкритим вихідним кодом. Він містить середу CoreCLR - кросплатформену реалізацію CLR. Нова платформа розвивалась і розвивається досить стрімко. Уже 16 листопада 2016 року був випущений .NET Core версії 1.1 [5], а 14 серпня 2017 року відбувся реліз .NET Core версії 2.0 [6

Треба зауважити, що створити .NET застосунок можна за допомогою декількох мов програмування: C#, F#, чи Visual Basic.

II. Постановка проблеми

На сьогоднішній день постає проблема створення зручного, багатофункціонального та, головне, кросплатформного застосунку. Адже все більшої популярності набирають програми, що можуть працювати більш, ніж на одній апаратній платформі і(або) операційній системі [3].

Кожна команда під час розробки стикається з необхідністю підтримки високого рівня якості свого продукту. Незважаючи на значну часозатратність, забезпечення якості продукту є невід'ємною складовою його успіху в майбутньому [3]. Зрозуміло, що можна розробити застосунок під одну платформу, але його вартість та кількість кінцевих користувачів буде значно нижча [1].

Довгий час платформа .NET надавала широкий ряд можливостей по створенню серверних, десктопних чи веб-застосунків, застосунків хмарних обчислень (англ. cloud computing) за допомогою хмарної платформи Microsoft Azure та застосунків для мобільних пристроїв. Але при цьому платформа .NET все ж таки не була кросплатформною та підтримувала тільки операційні системи сімейства Windows. Таким чином виникла потреба в розробці нової платформи, нової технології .NET, що розрахована на роботу з різними операційними системами.

Мета даного дослідження – переконатися в можливості розробки кросплатформних застосунків засобами .NET, дослідити інструменти розробки програмного забезпечення мовою C# чи їй подібних, переваги .NET Core.

III. Основна частина

.NET – це стандарт ECMA, який має різні реалізації: .NET Framework, Unity, Mono і тепер .NET Core. Тобто чимала частина функціональності є спільною для .NET Framework і .NET Core [7]. Але в нову реалізацію, .NET Core, закладені і дещо інші принципи.

Наступні характеристики найкраще описують .NET Core:

- Сумісність: .NET Core сумісний з .NET Framework, Xamarin і Mono за допомогою .NET Standard
- Інструменти командного рядка: всі сценарії продукту можуть бути виконані в командному рядку
- Відкритий вихідний код: платформа .NET Core відкрита, вона використовує ліцензії MIT та Apache 2. Документація ліцензується під CC-BY. .NET Core є проектом .NET Foundation
- Підтримка та супровід платформи корпорацією Майкрософт та спільнотою .NET у GitHub
- Кросплатформеність: працює на Windows, MacOS та в деяких дистрибутивах Linux. Додавання більшої кількості підтримуваних операційних систем (ОС), архітектури процесорів та сценаріїв застосування - основне завдання корпорації Майкрософт, яка ставить собі за ціль можливість .NET Core працювати на максимально можливому діапазоні пристроїв та операційних систем
- .NET Core має фундаментально модульну архітектуру. Runtime, різні бібліотеки, компілятор та інші компоненти спроектовані таким чином, що їхня взаємодія відбувається за допомогою інтерфейсів, а вони самі виступають як окремі сутності.
- Гнучке розгортання застосунків .NET Core

Однією з ключових переваг .NET Core є портативність. Код, написаний на .NET Core можна налаштувати для виконання на різних підтримуваних платформах. В залежності від налаштувань в ваших проектах код, що використовує .NET Core, може працювати на платформах .NET Framework, Mono та Xamarin, в Windows 8 та Windows Phone, а також на Universal Windows Platform (UWP). Ви можете хостити кілька застосунків одночасно, використовуючи різні версії CoreCLR, і окремо їх оновлювати, тобто, ви не зобов'язані оновлювати їх одночасно.

Нарешті, платформа .NET Core буде продуктивною. Одна з цілей .NET Core – зробити витрати кожної абстракції зрозумілими розробникам за допомогою реалізації моделі «плата тільки за те, що використовується» (pay-for-play model), яка робить очевидними витрати від застосування абстракцій вищого рівня для вирішення якогось завдання. Крім цього, .NET Core буде працювати максимально продуктивно із стандартною бібліотекою, яка мінімізує операції виділення пам'яті і загальний обсяг пам'яті, що займає ваша система.

Насьогодні існують чотири сценарії .NET Core: кросплатформні веб-застосунки ASP.NET (ASP.NET Core), кросплатформні бібліотеки та інфраструктури, кросплатформні консольні застосунки, а також UWP-застосунки.

Тепер є можливим розгортання власного веб-застосунку ASP.NET на Linux. Тобто ASP.NET Core – це новий кросплатформний веб-стек для .NET Core.

Відмінність між кросплатформними бібліотеками та інфраструктурами полягає в ступені масштабування.

Нарешті, UWP-застосунки, які довгий час були орієнтовані на сімейство пристроїв з операційною системою Windows 10, тепер можуть виконуватися в .NET Core.

Говорячи про кросплатформеність, не потрібно забувати про інструменти розробки. Зараз ми можемо обрати один з таких інструментів: Visual Studio, Visual Studio Code, Visual Studio for Mac, JetBrains Rider.

На даний момент Visual Studio є досить потужною IDE. Це найкращий вибір серед розробки на Windows машинах, тим паче є її безкоштовна Community версія [8].

Visual Studio Code є чудовим кросплатформним редактором. До того ж він підтримує чималу кількість мов та технологій програмування. Редактор працює на Windows, Mac OS та Linux, підтримує підсвічування синтаксису, є Intellisense – це загальний термін для різноманітних функцій редагування коду, зокрема: завершення коду, інформація про параметри, списки членів типу тощо, а також існує величезна кількість різних плагінів і розширень.

Visual Studio for Mac – це повноцінна IDE. Як і в Windows розробники, що полюбляють Mac OS, мають змогу крок за кроком обирати необхідні параметри для створення проекту. У Visual Studio for Mac є всі переваги сучасної IDE. Також є в наявності безкоштовна Community версія.

JetBrains Rider також є повноцінною IDE. Але її перевагою є те, що вона працює на Windows, Mac і Linux. Працює швидко, підтримує більшість видів .NET проектів, зокрема десктопні, web-застосунки, бібліотеки, підтримує Unity і Xamarin та, звичайно, .NET Core. Недоліком є те, що дана середа розробки є платною.

IV. Висновки

У статті було доведено можливість кросплатформної розробки програмного забезпечення засобами .NET, підтверджено актуальність нової технології .NET – .NET Core. Були розглянуті найбільш популярні та зручні інструменти розробки програмного забезпечення для різних операційних систем. Робота може використовуватись розробниками під .NET платформу, які вирішили зайнятися кросплатформною розробкою. Адже у статті наведені освновні переваги .NET Core.

Список використаних джерел

[1] URL: http://lib.mdpu.org.ua/e-book/vstup/L4.htm - Лекція 4. Кросплатформність. Види і типи сучасних мов програмування.

[2] URL: https://dou.ua/lenta/articles/net-core/ - .NET Core: возможности и перспективы.

[3] URL: https://dic.academic.ru/dic.nsf/ruwiki/989950 - Кроссплатформенное ПО.

[4] URL: https://arstechnica.com/information-technology/2016/06/net-core-1-0-released-now-officially-supported-by-red-hat/ - .NET Core 1.0 released, now officially supported by Red Hat. Ars Technica. Condé Nast (27 июня 2016).

[5] URL: https://blogs.msdn.microsoft.com/dotnet/2016/11/16/announcing-net-core-1-1/ - .NET Blog (16 November 2016). Announcing .NET Core 1.1.

[6] URL: https://blogs.msdn.microsoft.com/dotnet/2017/08/14/announcing-net-core-2-0/ - .NET Blog (14 August 2017). Announcing .NET Core 2.0

[7] URL: https://docs.microsoft.com/ru-ru/dotnet/standard/choosing-core-framework-server - Choosing between .NET Core and .NET Framework for server apps

[8] URL: https://www.visualstudio.com/downloads/ - Downloads | IDE, Code, & Team Foundation Server | Visual Studio

Ways of data exchange between sessions and scenes in Unity3D

Yulia Chukhrii
Computer Systems and Networks Department
Scientific adviser:PhD, assoc prof Chebanyuk O.V.
Software Engineering Department
Institute of Computer and Informational Technologies
National Aviation University
Kyiv, Ukraine

***Abstract*—Ways of data transmission between game session and scenes are considered in this paper. Such ways as serialization, player preferences, and external data storage are discussed.**

Keywords—Unity3D, game development, Serialization, DataBase, dreamlo, PlayerPrefs

I. INTRODUCTION

Modern video games have a large amount of data, such as score, time, place where the player stopped and other. So much information requires effective and rational approaches of management of data for saving computer memory and reducing the weight of game. Each of the methods listed below represents the ability to manage data to achieve an optimal game performance.

II. WAYS OF DATA EXCANGE BETWEEN SESSIONS AND GAME SCENES IN UNITY3D

A. *PlayerPrefs approach*

PlayerPrefs is a class in UnityEngine which stores player preferences and accesses to it. This way is effective to manage a small volume of data. PlayerPrefs has a key (PlayerPrefs.HasKey(“ ”)) and the value [1]. The value which is stored there can be obtained in such ways [3]:

- PlayerPrefs.GetInt();
- PlayerPrefs.GetFloat();
- PlayerPrefs.GetString();

To save a new value the next functions should be used:

- PlayerPrefs.SetInt();
- PlayerPrefs.SetFloat();
- PlayerPrefs.SetString();

To delete a key from the preferences the next method PlayerPrefs.DeleteKey(); is valid. To delete all keys and values corresponds PlayerPrefs.DeleteAll().

The example of usage PlayerPrefs is shown on the fig.1. PlayerPrefs.SetInt (“material”, ApplyChar.counter) receives a key “material” and a value ApplyChar.counter which is integer. It saves the properties of the object which was chosen by pressing “Apply”. Then Application.LoadLevel() loads the level that has a name “ig” where properties of chosen object will be delivered.

```
public void OnMouseDown()
{
    press = true;
    if(text.text == "Apply")
    {
        PlayerPrefs.SetInt("material", ApplyChar.counter);
        Application.LoadLevel("ig");
    }
}
```

Figure 1: Example of usage PlayerPrefs in the method OnMouseDown()

B. *Serialization*

XML, JSON and BSON: XML is type of data represented in textual form. XML is able to move data across the different port, but it required encoding when it passed over HTTP. This makes data communication more difficult due to the large size of data messages.

1) JSON (JavaScript Object Notation) is a technique of representation the data in text form which is human readable and it can be easy passed through without any message encoding that makes JSON more convenient to transfer data between the client and server. Today JSON is highly used so this format of data is supported by mobile applications and REST services. However, it is not natively supported by .NET language such as C# [5]. Therefore it is necessary to use JSON.NET (a JSON framework for .NET).

2) BSON (Binary JavaScript Object Notation) is a binary representation of simple data structures, associative

array, called a document. Documents (or objects) consist of an ordered lists of elements. Each element has a field name(string), a type and a value. It can store binary data directly. This eliminates the additional encode overhead but it gave a slight disadvantage in a space efficiency, because it has overhead for field names in serialized data.

Built-in features of serialization: Serialization is an automatic process of transforming data structures into a format that Unity can store and reconstruct. Some of the built-in features of Unity use serialization automatically, such as saving and loading of data, inspector window, prefabs and instantiation [2].

Saving and loading: Serialization is used to save or load scenes and assets to or from computer`s hard drive. This includes data saved in your own scripting API objects, for example, MonoBehaviour components or ScriptableObjects.

Inspector window: Unity uses serialization to display the value of GameObject in inspector window but it does not communicate with the Unity Scripting API when it displays the values of a field.

Prefabs: Prefabs are serialized data of one or more GameObjects and components. A Prefab instance contains a reference to the Prefab source and a list of modifications to this Prefab. From these two sets of serialized data Unity Editor instantiates a GameObject during the project build.

Instantiation: When the method Instantiate is called Unity serialized anything that exist in scenes such as prefabs and GameObject. Then Unity creates a new GameObject in which data will be deserialized.

Custom serialization: In cases when Unity serializer can not transform something that should be serialized, the concept of callbacks (ISerializationCallbackReciever, the interface in UnityEngine) will be actual. Callbacks allow hard-to-serialize data to be represented a different format which Unity can understand [4]. All data should be transform in Unity-understandable type before it wants to serialize this. Later, it can be transform into former form. ISerializationCallbackReciever only works with clases. Actual public methods for callbacks are:

- OnAfterDeserialize – to receive a callback after Unity deserializes the object.
- OnBeforeSerialize - to receive a callback before Unity deserializes the object.

3. Data storage in external resources

One of the ways to storage data in external resources is an online servers. Dreamlo is one of many representatives of servers that use a database for storage. The communication, changes and updates in database are done by HTTP GET requests using private URL. Dreamlo generates individually for the user a private, public code and URL which is necessary to paste in the sample code for Unity (fig.2). To create an access to the web page and get data from web server the class WWW is used. There is also possibility to stream or load a new player data files.

WWW has such methods [6]:

- GetAudioClip - to get an audio data (read only).
- EscapeURL – to escape characters in a string to ensure they are URL-friendly.
- Dispose – to dispose of WWW object.

WWW has such properties:

- progress – reports a status of download.
- isDone – is the data already downloaded?
- error – to receive an error message if there was an error during the download.

Figure 2: Example of public and private code generated by dreamlo

III. Conclusions

Through the work, the ways of Data exchange between sessions and game scenes in Unity3D were considered. In cases when the game has a small amount of data the method PlayerPref is most appropriate.

When the project has a voluminous data which have to be transferred from the client to server most quickly or Unity can not recognize the format of data, the way of serialization is used.

If the game is multiplayer and has many tables or leaderboards that require synchronization with external databases, the method of exchange and storage data on the web servers is up-to-date.

REFERENCES

[1] J. Hocking, Unity in actions, pp.352, June 2015.

[2] G. Lukosek, Learning C# by Developing Games with Unity 5x, 3rd ed, pp.230, March 2016.

[3] Unity documentation – PlayerPrefs – https://docs.unity3d.com/ScriptReference/PlayerPrefs.html.

[4] Unity documentation – Script Serialization – https://docs.unity3d.com/Manual/script-Serialization.html.

[5] Serializing data into XML, JSON and BSON – http://www.dotnetcurry.com/csharp/1279/serialize-json-data-binary.

[6] Unity documentation – Scripting API – https://docs.unity3d.com/ScriptReference/WWW.html.

A tool for designing architectural solutions in a cloud environment

Maksym Chukhrii
Scientific adviser: PhD. Assoc. Prof. Chebanyuk O.V.
Software Engineering Department
Institute of Computer and Informational Technologies
National Aviation University
Kyiv, Ukraine

***Abstract* — This article describes a proposal of an approach of transforming requirements described in UML to infrastructure model and description for different Platform-as-service (PAAS) and Infrastructure as Service (IAAS) in several public cloud computing providers.**

Keywords—cloud computing, UML, Platform-as-service (PAAS), Infrastructure as Service (IAAS), Infrastructure as Code (IaC), MDE

I. INTRODUCTION

Today pubic cloud computing become major trend in IT [1]. With Platform-as-service (PAAS), Infrastructure as Service (IAAS) and Infrastructure as Code (IaC) gaining traction on IT market and mass migration of business in cloud [2].

Now IaC tools allows infrastructure itself to become defined and managed code [3] which means that it and can be analyzed, modeled, visualized and generated using standard notions such as UML and with aid of MDE tools.

II. FORMULATION OF THE PROBLEM

The modern MDE tools allow to model different aspects of software architecture.

Now since we can describe infrastructure as code we can apply MDE methodology to coded infrastructure as well. This will allow it to effectively analyzed using models. Unfortunately, there is no such method or transformation of existing models that will allow to generate infrastructure code neither there is an way to take existing infrastructure code and transform it to standard process-able model such as UML.

III. DESCRIPTION OF PROPOSED METHOD

The proposed method to solve this problem is following. The system's input data is cloud provider services description defined in XML format and system description in form of component model. Having this data, we can perform code generation for IoC tools (for example Terraform, Ansible). The code generator generate code analyze model's components attributes and compare them to cloud services described in XML document and generate code using predefined templates [4], [5].

IV. USED TOOLS

To preform creating and validation of models IBM Rational Rose is used.

Rational Rose is an object-oriented Unified Modeling Language (UML) software design tool intended for visual modeling and component construction of enterprise-level software applications. In much the same way a theatrical director blocks out a play, a software designer uses Rational Rose to visually create (model) the framework for an application by blocking out classes with actors (stick figures), use case elements (ovals), objects (rectangles) and messages/relationships (arrows) in a sequence diagram using drag-and-drop symbols. Rational Rose documents the diagram as it is being constructed and then generates code in the designer's choice of C++, Visual Basic, Java, Oracle8, Corba or Data Definition Language.

V. CONCLUSIONS

The proposed method can be implemented in existing MDE tools such as Modellio or EMF and can help to create more stable and manageable cloud infrastructure.

REFERENCES

[1] Cloud computing: the new frontier of internet computing - ieeexplore.ieee.org/abstract/document/5562494/
[2] Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS - http://ieeexplore.ieee.org/abstract/document/5557962
[3] Infrastructure as Code - https://link.springer.com/chapter/10.1007%2F978-1-4302-4570-4_9
[4] Algorithms for cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds - https://www.sciencedirect.com/science/article/pii/S0167739X15000059
[5] Efficient service recommendation system for cloud computing market - https://dl.acm.org/citation.cfm?id=1656078

The present and future of Android development

Duchkova Krystyna
Scientific curator: Tkachenko O. A.
Software engineering chamber
Institute of computer and Information technologies
Kiev, Ukraine
Kduchkova3108@gmail.com

***Annotation* — The work is devoted to the consideration of the problem of flaring android industry as a whole and Android applications. The work identifies the potential ways for the development of the industry and the prospects for work for young professionals.**

Keywords—Android; indastry; development; mobile

I. Introduction to a problem

Nowadays around 1 billion of people are using android in the world. Android was first launched in the year 2005. The Android mobile operating system is currently developed by the Google. Mainly the Android is open source. It can be installed on devices of various types. The app development is also becoming a big challenge for the mobile app development companies in Florida, New York and all over the world as the mobile operating systems especially the Android has been continually evolving. With more than 80 percent market share, Android is the dominant mobile operating system today. It's running on countless models of smartphones and tablets, as well as many other devices. Judging by this, one would think that programming for Android is simple and easy. Or is it? Problems are everywhere, and Android is not exception [1].

1) Buggy IDEs - have you ever tried to repair your car with a shovel? Or tried to pick up girls while driving your grandfather's 40-year old Yugo? In the Android world, we had an official IDE for Android development - Eclipse, which had a ton of problems and could drive you mad in 10 minutes. The Eclipse ADT plugin was just buggy, slow and unfriendly for more complex projects. We quickly got sick of it and were praying for a miracle.

2) OS fragmentation - Gingerbread (2.3.7) occupied quite a market share (at least 15-20 percent) of Android OS versions. As you already know, Android underwent complete overhaul with the version 4.0 (Ice Cream Sandwich) - we got new UI elements, new APIs for device hardware, new screen densities... This resulted in us having to be careful to optimize and program our apps to work well on the new as well ancient versions of Android. All this greatly affected our development process and resulted in prolonged development time with more bugs and crashes.

3) Slow emulators - We need to test our apps on different Android OS versions and screen dimensions, so we have to buy at least 20 different Android devices. Sounds crazy? OK, so we can use emulators. But have you ever tried to use the default Android emulator? It's so painfully slow that you'll soon catch yourself counting cars parked in front of your office while your app is being deployed to your emulator.

4) UI - Android apps were boring. If you commit blasphemy and take a look at iOS apps, you will see that they are full of life and colors. Everything is animated, transforming, going from left to right, right to left and so on... Our apps were static, and if we wanted to enrich our UX, the old Gingerbread would have soon killed all our hopes and wishes [2].

II. Main part

Android developers need to keep in mind when developing the android app, the app should be compatible with all latest devices. Nowadays in the mobile market there are different models are introduced, it is obvious that the some devices are outdated. So, when developing apps for those devices, it is more risky to the developers, so that the developers need to know changes in that and they should be updated.

Android as stated by Google, the entire Android environment has changed with the Lollipop. Android Studio is the new software introduced it has got popular in the market.

The project automation tools that Google introduced for Android developers helps them to separate their apps into various parts, assign proper configuration settings, and increase app's execution speed [4].

Material design is becoming more popular with Lollipop, which is added in the design of the UI of the operating system. It fully redefines the user experience. It shows the complete mission and vision it has about the future Android user experience, as stated by Google.

Now the smart watches are introduced to the market. It is also helpful for android development professionals. They can show their agility in smart watch app development with the help of the advanced operating system. Android is added with the latest features to support smart watch app development. In the future the smartphones may be used to operate the television, refrigerator, washing machine, which the android already has tried and tested for the applications in the future.

The Android application development is ready to move with newer technology and devices.

The flexibility and the scalability as well as the wide range of compatibility that the Android offers shall remain positive. The developers are getting the new way of working with smartphones with the help of internet. The Android is becoming more popular, and the Android is fully prepared to handle the future technology innovations [3].

III. Conclusions

Android is becoming more and more popular in the global market and it has introduced new features and adopting the new technologies to attract more and more existing and new customers. It has given the lot of opportunities to Android developers. Android Development is having a brighter future by supporting the upcoming technologies.

A lot has changed in the past few years for Android. It has evolved from a simple OS for smartphones and is now powering many other devices. Time will tell what will become of it.

References

[1] Kroah-Hartman, Greg (December 9, 2010). "Android and the Linux kernel community". Linux kernel monkey log. Retrieved June 20, 2017.

[2] Paul, Ryan (February 24, 2009). "Dream(sheep++): A developer's introduction to Google Android". *Ars Technica*. Condé Nast. Retrieved June 20, 2017.

[3] "Google's Android OS: Past, Present, and Future". PhoneArena. August 18, 2011. Retrieved March 12, 2017.

[4] "What Are The Major Changes That Android Made To The Linux Kernel?". Forbes. May 13, 2013. Retrieved June 20,2017.