PhD Progress Report 2

Lorand Kedves
2017/2018. I.

Tensegrity in Software Development

Lorand KEDVES

Doctoral School of Information Science and Technology
University of Pannonia
Supervisor: Dr. Botond BERTÓK

Summary

This PhD research emerged from 20+ years of experience in software development and maintenance, requirement analysis and system architecture design: how to extract and emphasize the structure and configuration of any software, favoring an abstract, visual form against thousands of lines of code, and how far this approach can reach. For scientific closure: is it possible to achieve total coverage, when all necessary source code, configuration or binary can be generated directly from the design?

The progress in the past 2 semesters was significant. In active projects, I have overcome some of the limitations of previous experiments, and succeeded in integrating new areas: change management, a better graph representation and connection to binaries. The academic research revealed solid background of the initial aim: experiments in 1960-80 and the outstanding achievements of Douglas Engelbart; and interesting connections to parallel areas: Tensegrity architecture concept from Buckminster Fuller, and knowledge representation: critical thinking and argument mapping. The courses taken also supported the ideas, and an article is under review at an international journal.

Activity and Experience

Follow-up on the Previous Report

Efficient Coding – WYSIWYG Editor

The editor is accepted by the Government, we are now in the first round of real life usage and corrections/improvements. The fundamental concepts and results:

- Forget about standard OOP concepts like “represent everything with Java classes”, and use a far more flexible Map-based solution;
- Clean MVC separation (you can’t mix Model with Control if you don’t have a Java class representing the “thing”), by late attachment of Control objects;
- Distinguishing “validators” (direct individual reaction and possible rejection/correction of user activity) from “agents” (post processing changes of perhaps hundreds of changed entities only once instead of doing it one by one) gave a great performance boost and cleaner system structure;
- This resulted an almost complete Undo/Redo support (legacy components and late changes had negative effect, but the minor limitations were accepted).

I currently give lectures at the company about coding concepts and their application in the system, and also participate in defining the next milestones.

Knowledge Card Management

I have recommended a knowledge collection mechanism to a pharmaceutical company to organize their actual knowledge of one of their IT systems before selecting a new solution. They have completed their survey using "Knowledge Cards" in Word, but we also started to plan a manager software.

I found that I just re-invented the knowledge management mechanism that Douglas Engelbart used and described in Augmenting Human Intellect report in 1962. This idea was implemented as Hypercard manager by Apple, as presented in this video from 1987. I could not write anything comparable to a commercial product by Apple, so the development plan was pointless. However, it is another example of my opinion that the current course of IT, both industrial and academic, has lost its creative and scientific momentum, and previous results, so we waste lot of time on repeating them.

Research in Computer Science History

In the past report I listed Alan Kay, Ivan Sutherland, Seymour Papert and Marvin Minsky. Since then I have changed my focus to another path: design and implement the information technology that is able to support human intellect in a cooperative way. The need of such tools is presented by Vannevar Bush in his ground-breaking article, As We May Think (1945). The interface, services, estimated performance and timeline is described by J.C.R. Licklider in the Libraries of the Future book (1964). The means and results of augmenting human abilities (from physical to intellectual tools) were thoroughly analyzed by Douglas Engelbart; I have processed Augmenting Human Intellect report (1964), the Mother of All Demos presentation (1968), several lectures about the A-B-C Model, "Bootstrapping" and collective IQ.

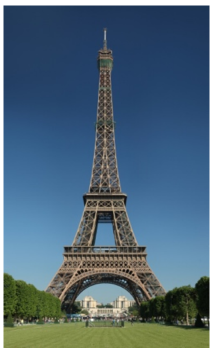
Related Research Areas

Tensegrity

Civil engineering and architectural design is commonly used analogy in software development, although it inherently lacks the dynamism of real life information systems (where the requirements change both during and after building the system). We try to type enough code to create service objects (like collections, UI widgets, DB connections, etc.), connect them and define their cooperation that provides the required behavior. However, the mass of code and the decisions we have to make at the beginning to have a running system hit back later: the solutions becomes rigid, and as it advances, harder and harder to change.

In the '60, Kenneth Snelson and Buckminster Fuller recommended a radically different approach, where the structure should follow the actual tension and compression forces, and never connect rigid, compression resistant elements directly, but through pre-stressed tension resistant elements. These structures consist of minimal amount of material, can distribute external forces along the whole system, can grow or change more organically when needed. This model could make software development more efficient, and is a direct analogy of the Dust Platform structure.

Without Architecture
(building blocks held together by gravity and friction)


Traditional Architecture
(Compression and tension in blocks and at artificial joints)

Tensegrity
(Compression and tension resistent components separated)

Philosophy, Sociology

Information technology is tools and methods that allow us, human beings to deal with more information, with greater precision and less physical limitations. This inherently changes how we learn, reason, create and express our opinion, and all the examined researchers were conscious about this fact. Bush and Licklider expressed their serious concern about this change, and gave surprisingly precise predictions about the possible dangers and our current problems related to it. Douglas Engelbart gave a precise model of how this human-machine cooperation can lead to radically more efficient individual and group operation. Then he spent the remaining 50 years of his life trying to explain how organizations can improve using his A-B-C Model with no significant result, which leads to his statement: "The computer revolution has not yet started".

Consequentially, working with these concepts lead out of standard technology environment. This is why I still find the lectures from Neil Postman and Ted Nelson extremely important. This is why Engelbart's concepts are referred to by Tim van Gelder (University of Melbourne, focusing on informal reasoning and critical thinking). Following this path, I plan to work on materials from Dona Warren (University of Wisconsin, research area is Argument Mapping).

I think the Dust Platform and the concepts behind are in direct confrontation with the contemporary state of the art of information science and industry, so my first "short note" article was considered being out of scope by Information and Software Technology journal. It is likely to get more welcome in this more abstract area, if I can cooperate in building knowledge representation, ontology or argument mapping tools.

Open Source Civilization

Based on my results, I have participated in "The Global Challenges Prize 2017: A New Shape – Remodelling Global Cooperation" with a 7000+ words proposal, Open Source Civilization. This article explains

- why our current corrupt understanding is the fundamental cause of our problems;
- how the mentioned scientists predicted this situation and why the envisioned computer revolution is essential element of such a "new shape";
- where Dust Platform can help in a better cooperation.

The proposal is private during the decision process, later I plan to publish it.

Dust Platform

The current version of Dust Platform profited much from the above-mentioned research and experience.

- The over-complicated identification mechanism is removed completely, there is no real need of a globally unique naming scheme. To build a system, it is enough to collect the used entities and create relations among them, or use tagging for system-wide identification. For more global access, the Source components (currently JSON files or database connectors) provide identifiers.
- The simplest representation is achieved: the JSON file only contains an "entities" and a "relations" array; all aspects of the entity became equivalent, including the "entity" aspect. At the same time, generic attributes (Boolean, numeric, string) became "fields"; multiple values are not allowed (handled by a "Container" in relation with the owner).
- The connection with business logic and function call is further developed by Service (interface), Command (function), LogicAssignment (connect a Java class to a service) types and the Services relation (connects Services to the Entity).
- A standard application (a resource management system in Java; with servlet container center, web and mobile clients; MySQL database persistence; standard user authentication and authorization) is under development based on the Platform.

Academic Results

Publication

The findings and conclusions from the research were summarized in a 2500-word short note article: "The Science of Being Wrong - Unsung Heroes of Informatics". It is under review at Technological Forecasting & Social Change (international journal with impact factor 2.6 in 2016). Further publications are planned discussion various aspects of Dust Platform, like experiences in representing real-life software in graphs, ontology and knowledge management, etc.

Courses

During the first semester, I was able to complete 2 courses versus the originally planned 4. The remaining 2 was taken again.

Analysis and Synthesis of Technical Processes – Dr. Botond Bertók

We focus on the generalized software modelling, ontology management, platform independence, quality, maintenance and performance issues. We use the mentioned editor software as a use case, and the current implementation of Dust Platform to validate the requirements and efficiency of the selected approach. Later on, we evaluate using this concept in academic environment: store and improve local knowledge bases.

I have also given a lecture to correspondent MSc students with the title If You Have a Hammer… why and how I created a new software platform. I presented

- what we lose when translating the component network structure of any software to a certain programming language and let its runtime execute it;
- what can be achieved if we have the runtime in our hands;
- and why is it possible that big companies have not addressed this area.

I also recommended keeping an eye on the questions of any software development in this order:

- Why (you need this component, and have to do it yourself instead of using an existing solution)?
- What (subcomponents, parameters and relations are necessary to perform the operation)?
- How (can you actually implement that structure on the selected platform language and tools)?

Graph Coloring – Prof. Zsolt Tuza

During the past year, I got much closer to my original aim: a way to represent knowledge in general, and as a subset, any software in the form of entities (nodes) and relations among them (edges), where the entities can have dynamic attribute packages ("types") while the relations' attributes are fixed: the relation type and optionally, an index. To execute such a graph, I only need some fixed types and relations (like: LogicAssignment, Application, Sequence, etc.) I already have another module that allows describing the types, entities and relations using the same tools, therefore, the representation is self-containing.

The mathematical equivalent of this structure is the colored graph: an interconnected network of nodes where both the nodes and edges can have attributes assigned (in the simple form, one color, but in more complex representations like Petri nets, different attribute groups). Besides overviewing the theoretical background of this area, we should ensure that the chosen representation method does not contain anything not covered by the mathematical definitions. If that holds, graph algorithms can be directly applied to anything represented in Dust, either an ontology, or a software.

Veszprém, 2018.01.10

Lorand KEDVES	Dr. Botond BERTÓK
PhD Student	Supervisor