Research Proposal

Field of research
Computer Science/Information modelling

Topic of Research
Information modelling for software services

Background of Research

The problem of interoperable and easily accessible services and interfaces is among the core problems of modern computation. There are numerous attempts to model and abstract them in operating systems-level, such like files, executables, Component Object Model (COM), POSIX, RESTful, and a lot of related supporting models like semantic file system, aspect-oriented and component-based development.

These years with the prevalence of Web and mobile computing, the unsolved gaps between services and APIs which are reflected by almost-forgotten attempts like Common Object Request Broker Architecture (CORBA) are still there, partly shed by the increase of computation capacity.

For end users, they require transparent, out-of-box and friendly solutions. They tend to ask: Why I still bother changing file formats for the data with same information (such like spreadsheets, an audio)? Why there are so many software products but to find the suitable (and sometimes niched) functionality among behemoth products it is often like to look for a needle in the haystack?

For software developers, they require manageable interfaces and suitable abstractions. They tend to ask: Why I am still bothered by running environment issues such like “dependency hells” (DLL hell, JAR hell, etc)? Why cross-platform is still so difficult that you cannot imagine send a computation task to another machine and run it without hassle?

The stem of these problems is the information model of software services are still unclear, at least no commonly accepted model is used. The software development therefore cannot implement simple principles in programming languages, such like inheritance and function composition, but should have active usage of information models.

Problem Statement

1.Refinement and development of information models for software services.
1.1. How to model different granularities of software services, from simple arithmetic in single machine to concurrent processes among nodes.
1.2. Frameworks for reconciliation of different platforms, such like Windows desktop, Android mobile platform, Linux servers. This is related to binary compatibility such like Java or “dynamic” compatibility such like On-the-fly code generation.
1.3 Key theoretical problems related to concurrent implementation Lambda Calculus. Planned works on semantic aspects or implementation including referential transparency and problems.

2. Development of user-friendly product
2.1 Natural language – formal language interface for task invocation and service positioning.
2.2 Auxiliary tools to manage services, programs and machines with strong integrity.

Objective

To provide better understanding of the computation models.
To improve the framework for software compatibility.

Related researches and literature review

In Lisp machine and Oberon, there is no separate shell language like bash in Unix or Powershell script in Windows. The shell language is REPL of the language (Lisp-machine Lisp, and Oberon) the operating system is implemented in. The “programs” installed in the OS are actually small but easily manageable program pieces which can be called and reused easily.

Lisp machine suffered from being bloated after introduction of the message passing framework Flavors, and became by nature similar to thread-based systems. The evolution of Oberon on the other hand, experienced adoption and abandon of object persistence and kept thread-like Active Object.

Saltzer, J.H., 1978 ’s highlighting of naming and binding of object in OS is still informative in nowadays situation. My inspiration from his and related research is that, the design of the system should

In the recursive network model proposed by John Day 2007, the abstraction is directly related to bandwidth of data. Suppose the computer system can only process data counted by bytes, then unstructured file is more suitable than semantic rich approaches, but higher abstractions like String, audio, images, video, contract and things in Internet of Values appear naturally as computation reaches a wider bandwidth and thus more complexity. This will serve as an exemplar for “recursive software model”.

Current exemplar of information and service interoperation is IFC (Industry Foundational Classes) and the area of application is Building Information Modellling(BIM). Within accepted frameworks, a user does not need to worry about the representation of physical objects, this success is partly due to semantic simplicity of building information, but when it comes to Facility Management which deals more human activities, the models are also limited. No models are perfect but in this research existing models from other industry areas like IFC model will be modified and extended to modify software services.

Component-based development is originated from McIlroy’s idea in 1960s and became with Web because being service-oriented is one of the idealist architecture of Web services. The techniques for component-based including dynamic linking and embedding, bytecode manipulation (for JVM and others virtual machines). Java Beans can be regarded as a successful product, but it is not an information model and many developers have crude mappings between Java and Java-like language features and information models.

Some existing software products can be list as:
Terraform: uses of DSL to perform tasks derived from batch processing systems and uses a graph-based model for task management.
RESTful: a number of status management frameworks in Javascript ecosysem such as Flux, Redux and Vuex.
Apache Spark and Storm/Kafka which abstracts data as mini-baths and streams respectively. (Lin, J., 2017)
“Intelligent shells” like ZSH which integrates some natural language-like interactions but it assumes full knowledge of services to use and even their internal structures.

However, the problems left to be solved include at least:
1. How information models can be used to improve software models?
2. Which stage, compilation, linking, or loading even runtime should information

Theoretical framework

Firstly, basic computational theories like Lambda Calculus are used. They are the root of problems, although look rather low-level. I hope semantic aspects of Lambda Calculus or LC-based theories can be developed on the basis of Church–Rosser theorem.
Secondly, various information models including Entity-relationship model and derived standards like STEP (STandard for Exchange of Product Model Data) will be used to bridge the gaps between pure computational process and human knowledge domains.
Thirdly, key techniques like tree-based program representation should be used and improved for implementation.

Methodology

1.Theoretical research, critiques.
2. Experiments and practical research, including programming and testing product models.

Schedule of research work

Theoretical study around 0.5 year, focusing on computational theories and modelling techniques and searching for related research.
The remaining of the time is devoted to building products and thesis writing.

Expected result

1. An information model for software services
1.1 The key issues of granularity are identified, analysed and better solved.
1.2 The feasibility of such a framework and its core components, represented by the model developed in 1.1.
1.3 An answer on problems raised for computational theories, especially Lambda Calculus.

2. At least a toy product can be used for single machine.
2.1 Natural language, formal language and GUI mixed user interface. To use “functors” (functionalities) different means can be combined.
2.2 Integrating traditional “files” and “programs” into semantic-rich objects, which is similar to unsuccessful “Library” abstraction in Windows File Explore but there are fundamental updates.

Ethics
No conflicts or major issues regarding ethics is found.

Budget
Exact amount is to be determined. There is no foreseeable requirement of purchase of hardware products but some expenditure on intelligence property including books, articles, software products is expected.

Conclusion
This research is targeted at and aimed to solve key problems of abstraction and encapsulation of components, be it an “object” or “service” with a high standpoint dubbed as “information model”.

References

Saltzer, J.H., 1978. Naming and binding of objects. *Operating Systems*, pp.99-208.
John Day 2007. Patterns in Network Architecture: A Return to Fundamentals
Lin, J., 2017. The lambda and the kappa. *IEEE Internet Computing*, *21*(5), pp.60-66.
Matthias Felleisen, Robert Bruce Findler, Matthew Flatt 2009. Semantics Engineering with PLT Redex
Peyton Jones, S.L., 1987. *The implementation of functional programming languages (prentice-hall international series in computer science)*. Prentice-Hall, Inc..
Chen, P.P.S., 1976. The entity-relationship model—toward a unified view of data. ACM transactions on database systems (TODS), 1(1), pp.9-36.
AL-Khawlani, M.M., 2004. Web services: a bridge between Corba and DCOM. *Journal of Science and Technology*, *9*.
Saltzer, J., 1993, August. On the naming and binding of network destinations. In *Local Computer Networks* (pp. 311-317).
Weinreb, D. and Moon, D., 1981. The lisp machine manual. *ACM SIGART Bulletin*, (78), pp.10-10.
McLay, M.J. and Morris, K.C., 1990, August. The NIST STEP Class Library. In *C++ at Work-'90 Conference Proceedings.*
Douglas A. Schenck, Peter R. Wilson, 1994. Information Modeling the EXPRESS Way
Atkinson, M.P. and Morrison, R., 1985. Procedures as persistent data objects. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, *7*(4), pp.539-559.
Alhazbi, S., 2009. *A Framework for Dynamic Updating in Component-based Software Systems* (Doctoral dissertation, UNIVERSITI SAINS MALAYSIA).
Weilkiens, T., Lamm, J.G., Roth, S. and Walker, M., 2015. *Model-based system architecture*. John Wiley & Sons.
ABRAHAM. JORIJ, 2016. *PRODUCT INFORMATION MANAGEMENT: Theory and Practice*. SPRINGER.
McIlroy, M.D., Buxton, J., Naur, P. and Randell, B., 1968, October. Mass-produced software components. In *Proceedings of the 1st international conference on software engineering, Garmisch Pattenkirchen, Germany* (pp. 88-98).
Stuart Dabbs Halloway, 2002. Component Development for the Java Platform
Wetherbee, J., Nardone, M., Rathod, C. and Kodali, R., 2018. *Beginning EJB in Java EE 8: Building Applications with Enterprise JavaBeans*. Apress.
Lévy, J.J. and INRI, A., 1988. Sharing in the Evaluation of lambda Expressions. *Programming of Future Generation Computers II, North Holland*, pp.183-189.
Mössenböck, H., 2012. Object-oriented programming in Oberon-2.
Szyperski, C., Gruntz, D. and Murer, S., 2002. Component software: beyond object-oriented programming.
Franz, M.S.O., 1994. Code-generation on-the-fly: A key to portable software.