CompReducer : tools for quickly and automatically create components to aid building resource adaptive software systems

Arpit Christi
EECS
Oregon State University
Corvallis, Oregon, USA
christia@oregonstate.edu

Alex Groce
EECS
Oregon State University
Corvallis, Oregon, USA
agroce@gmail.com

***Abstract*—The abstract goes here.**

I. Introduction

Modern day software systems are very complex consisting of various resources like libraries, models, operating systems, databases, memory systems, processors drivers, browsers, services, data structures etc. While developing such complex software systems, certain implicit or explicit assumptions are made for the resources that either software is going to use or the software is going to be operated under. For example, a mobile application using location assumes that some sort of location provider services is available via network or GPS or through some other mechanism. Advances in underlying technology, sometimes continuous or sometimes drastic forces software developers to adapt/evolve their software to these underlying changes. Users of such systems have to go through updates in software or sometimes buying new technology to continue to use certain software. A change in a library used by the software sometimes forces developers to refactor or rewrite part of their software which can lead to a multiple cycles of development and testing before the application can be fully adapted to the library changes, a costly, time consuming and error prone process.

Having a mechanism to design and implement resource adaptive real world software system that can adapt to changing environment or changing resources can solve the problem. A lot of work has been lately devoted to building self adaptive software system.s. In their survey of engineering approaches for self adaptive software systems Crupitzer et al discussed different approaches to build self adaptive software systems. The approaches they discuss include but not limited to model-based, architecture-based, reflective, programming paradigms, control theory, service oriented, agent based, formal modeling and verification based, machine learning based and others. After so much of significant work in this area, we have still not seen advent of any industry strength adaptive software system on horizon yet. In fact at the time of writing this paper, authors are unaware of even a single industry strength resource adaptive software system. In order to design and implement resource adaptive software system for mission critical software, DARPA launched Building Resource Adaptive Software System project, BRASS, that employs a new clean slate approach that aims to capture the relationship between computations and the resources they use and provide transformations that enable applications to adapt to resource changes without the need for extensive programmer involvement.

mds
August 26, 2015

A. *Subsection Heading Here*

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

II. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.