Reviewer: Sharon Choy

Paper Title: An Overview of the Scala Programming Language

Author(s): Martin Odersky, Philippe Altherr, Vincent Cremet, Iulian Dragos
Gilles Dubochet, Burak Emir, Sean McDirmid, StÈphane Micheloud,
Nikolay Mihaylov, Michel Schinz, Erik Stenman, Lex Spoon, Matthias Zenger

1) Is the paper technically correct?
 [x] Yes
 [ ] Mostly (minor flaws, but mostly solid)
 [ ] No

2) Originality
 [ ] Very good (very novel, trailblazing work)
 [x] Good
 [ ] Marginal (very incremental)
 [ ] Poor (little or nothing that is new)

3) Technical Depth
 [ ] Very good (comparable to best conference papers)
 [x] Good (comparable to typical conference papers)
 [ ] Marginal depth
 [ ] Little or no depth

4) Impact/Significance
 [ ] Very significant
 [x] Significant
 [ ] Marginal significance.
 [ ] Little or no significance.

5) Presentation
 [ ] Very well written
 [ ] Generally well written
 [x] Readable
 [ ] Needs considerable work
 [ ] Unacceptably bad

6) Overall Rating
 [ ] Strong accept (award quality)
 [ ] Accept (high quality - would argue for acceptance)
 [x] Weak Accept (borderline, but lean towards acceptance)
 [ ] Weak Reject (not sure why this paper was published)

7) Summary of the paper's main contribution and rationale

for your recommendation. (1-2 paragraphs)

The main contribution of this work is that it presents Scala language which fuses object-oriented and functional programming into a statically typed programming language. The Scala language is novel in its type system as it supports parameterization and abstract members for the purpose of abstraction. Additionally, Scala has flexible modular mixin-composition constructs which allows programmers to reuse new class definitions that are not inherited. Lastly, Scala has views (implicit conversion between types) which enable component adaptation in a modular way. Scala attempts to solve the external extensibility problem by having views which allow the programmer to augment a class with new members and supported traits (special form of an abstract class which does not have any value parameters for its constructor). The motivation for Scala is to address the shortcomings of programming languages that are used to define and integrate components; ultimately, Scala was created to address the limited support for component abstraction and composition in statically typed languages such as Java and C#.

The contribution of Scala is significant because it provides a set of constructions for composing, abstracting and adapting software components. This ultimately allows the language to become extensible enough so that users can model their domains in libraries and frameworks. Scalaís design is influenced by many different languages (e.g. Smalltalk, Beta, ML, OCaml, Haskell, Pizza, etc.). Thus, this paperís originality comes from its efforts to amalgamate research efforts in the area of software abstraction and component composition. Also, the paper argues that Scalaís class abstraction and composition mechanisms can be seen as the basis for a service-oriented software component model. This contribution is also significant as it would allow pluggable software components; and thus, Scala is able to allow simple assembly of large components that have many recursive dependencies. Ultimately, the application of Scala would result in a ìsmooth incremental software evolution process.î Overall, the paper was technical and its depth was appropriate and comparable to typical conference papers. Sufficient explanation was given to the presented examples; however, an overall rating of weak accept was given since the paper could have been presented and written more clearly (i.e. it was noted that there were a number of grammar mistakes).

8) List 1-3 strengths of the paper. (1-2 sentences each, identified as S1, S2, S3.)

S1 ñ This paper clearly outlines the features of Scala in a logical format (e.g. Scalaís resemblance of Java, has a uniform object model, is also a functional language, has abstraction concepts for both types and values,

has flexible modular mixin-composition constructors, allows decomposition of objects by pattern matching, supports XML documents, and allows external extensions of components using views).

S2 ñ The paper provides Scala examples which aid in comprehension of the Scala programming language (although the paper is not intended as a tutorial)..

S3 ñ The paper does a fair job in explaining how Scala is both an object-oriented and functional programming language at the same time. The background the paper provides for each feature of Scala is sufficient for reader comprehension.

9) List 1-3 weaknesses of the paper (1-2 sentences each, identified as W1, W2, W3.)

W1 ñ The paper postulates that ìscalable support for components can be provided by a programming language which unifies and generalizes object-orientated and functional programmingî; however, the paper fails to explain the intuition behind this statement and how unifying object-oriented and functional programming can result in better language support for component software.

W2 ñ Though the paper provides a description of Scala, it fails to acknowledge whether or not Scala is helpful in designing component software. The paper indicates that the only way to evaluate the usefulness of Scala is to apply it to an application; however, it does not give any empirical evidence (or indicate of future work) that Scala eases the designing of component software.

W3 ñ The purpose of Scala, as suggested by the introduction, is supposed to help in designing component software; however, in each of the sections describing the Scala language, the authors do not clearly provide a rationale of how the described feature of Scala contributes to solving the presented problem.