Chris A. Mattmann, Ph.D.
Jet Propulsion Laboratory, California Institute of Technology /
University of Southern California / Apache Software Foundation

Big Data Architecture: Fundamentals

Sponsored by the Jet Propulsion Laboratory and by the Keck Institute for Space Studies, California Institute of Technology

Outline

- Introduction to Software Architecture
- Styles, Patterns, Reference Architectures
- Architectural Modeling, Visualization
- Architectural Drift and Recovery
- Case Study: Grid Computing
- Conclusions

And you are?

- Chief Architect at NASA JPL in Pasadena, CA USA
- Software Architecture/ Engineering Prof at Univ. of Southern California
- One of original PMC members for Apache Nutch
  - predecessor to Hadoop
- Apache Board of Directors involved in
  - OODT (VP, PMC), Tika (PMC), Nutch (PMC), Incubator (PMC), SIS (PMC), Gora (PMC), Airavata (PMC)

Some notes

- Talk optimized for breadth, not depth
  - Depth can be found in my CSCI 578 course at USC on Software Architectures
    - http://sunset.usc.edu/classes/cs578_2014b/
- Encourage you to check the references at the end and feel free to ask questions
  - http://twitter.com/chrismattmann @chrismattmann
  - chris.a.mattmann@nasa.gov

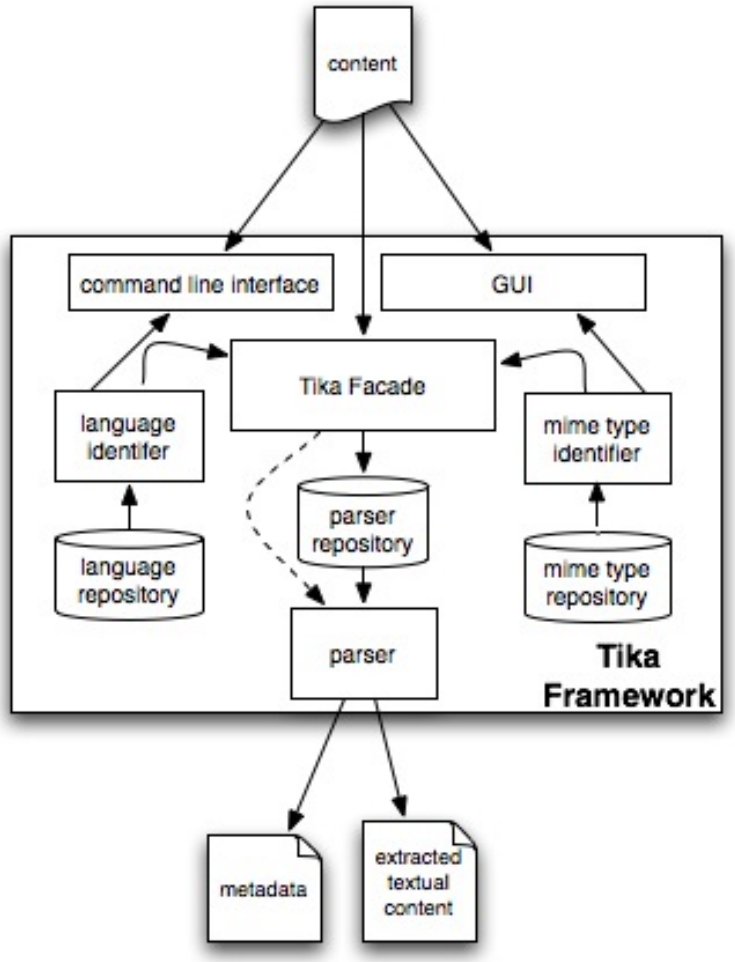
Software Architecture, An Introduction

- Analogies to building architecture
  - Materials in buildings are malleable, and software is not
    - "often the last kid on the block" in terms of updates
- Fundamental research
  - Perry and Wolf 1992 – Foundations for the Study of Software Architecture (Elements, Form, Rationale)
  - Shaw and Garlan 1996 – first appearance of "connectors" – Software Architecture Perspectives on an Emerging Discipline
  - Krutchen 1995 – 4+1 model view of Software Architecture – "aesthetics" emerges as a relevant principle
- Basic definitions (Medvidovic & Taylor)
  - Components – the units of computation in a system
  - Connectors – models of the interconnection amongst components and other connectors
  - Configurations – arrangements of components and connectors and the rules that guide their composition

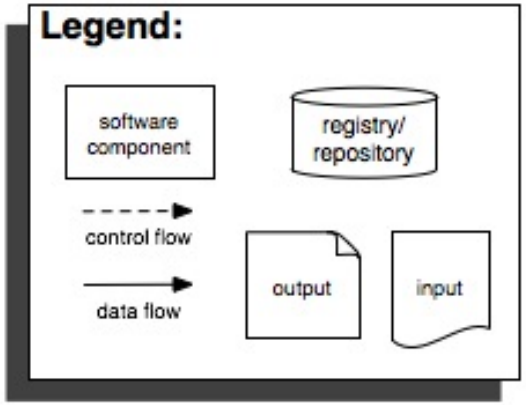
Core Definition

- Software Architecture
  - *"the principle design decisions about the software system" – Taylor, Medvidovic & Dashofy 2011*
  - What makes the decision "principle"?
    - Affects one of the core elements (component, connector, configuration)
    - Independent of implementation
    - Has to do with requirements
    - Has to do with system evolution
    - Others

Some Examples



Apache Tika: content detection and analysis toolkit, http://tika.apache.org/



What about it's:

- style?
- patterns?
- commonly used idioms and vocabulary?

Architectural Styles

- Architectural Styles
  - Commonly used components, and connectors and their *types*, intrinsic rules and guidelines
  - Think: *ingredients*
  - Common examples: *P2P, Client Server, REST, Layered* (we'll cover this in depth later)
  - Typically styles are identified and reused across domains after many systems are built in a domain
    - Common vocabulary, language, found successful

Architectural Patterns

- Architectural patterns
  - Lower level than styles, typically "code" patterns that constrain common arrangements of components, and connectors (configurations), as well as cardinality (number, size, etc.)
  - Leave the remainder as *explicit variation points* to be filled in by the instantiator
  - Think: *Recipes*
  - Common Examples: MVC, Three-Tiered, Sense-Compute-Control

Reference Architectures

- Reference Architecture, and Domain Specific Software Architectures (DSSA)
  - Originally identified by Will Tracz in DSSA paper in ACM Software Engineering Notes (SEN) - 1995
  - Three components:
    - Common Architectural Styles/Patterns
    - Reference Requirements (common requirements used across components and systems)
    - Domain model (common ontology and vocabulary)
  - DSSAs are reference architectures for a domain, e.g., avionics
- Examples: Apache OODT; Globus, Hadoop, etc., reference architectures in Big Data/Grid Computing