MAP Roadmap

v0.5
1 July 2008

MAP

- MIT Application Platform:
  - Software stacks: JEE (SASH), PHP (Zend), MySQL with associated toolkits
  - APIs to MIT infrastructure services
- Developer Tools:
  - Tools used by developers to create and manage code and processes

Vision

- Provide software stacks, APIs, toolkits and developer tools in order to:
  ① Lower the cost of SW development at MIT
  ② Produce better quality software
  ③ Rapidly develop SW in response to changing needs
  ④ Improve consistency and predictability
- Foster a developer community that is actively sharing tools, reusable code, and best practices
  – Requires buy-in from developers and managers

Goals

- Developers can build new applications from a toolkit of parts, rather than build all the components themselves every time
- Developers can integrate with IS&T's infrastructure services through appropriate interfaces
- SW projects have state of the art tools to facilitate best practices
- Infrastructure is in place to foster developer community
- MAP Working Group, Steering Committee are actively setting priorities and guiding development

Value to the Community

- Consistency of development practices and tools improves predictability
- Re-use of code and components improves efficiency of development cycle
- More uniform user experience
- MAP is community-driven, and therefore meets its needs

Trends/Drivers

- MIT's software infrastructure was very advanced 15 years ago – now it needs replacing
- Big SW projects that take forever to deliver are the past – needs and technology change too fast
- Student VISION is the meteorite on its way
- Service-oriented Architecture
- Rich Internet Apps

Current State: Stacks

- Assets:
  - SASH stack for Java
  - Working on a Zend/Drupal stack for PHP
  - JQuery for AJAX
  - MySQL cluster underway
- Gaps:
  - Largely determined by MAP working group, steering committee in response to ongoing and new work
- Concern: How do we respond to the changing world? (i.e. new stacks as needs change)

Current State: APIs

- Assets:
  - SOAP services with WSDL
- Gaps:
  - incomplete library of reference implementations and documentation for integration with our web services

Current State: Dev Tools

- Assets:
  - Source control (SVN)
  - Build Dependency management (Maven)
  - Continuous Integration (Bamboo)
  - IDE (MyEclipse)
  - Code browser (OpenGrok for Kerberos team,FishEye)
  - Issue management (Jira)
- Gaps:
  - Code analysis
  - Load and stress testing tools

End State: Stacks

- Stacks for Java, PHP
- Shared MySQL cluster
- MAP working group, steering committee help define priorities for new stacks in response to community needs
- Stacks used by Student Vision, other SAIS development projects, non IS&T developers

End State: APIs

- SOAP and REST APIs to MIT infrastructure services
- Complete set of reference implementations and libraries to access MIT infrastructure from Java and PHP
- Implementations as required for Student Vision

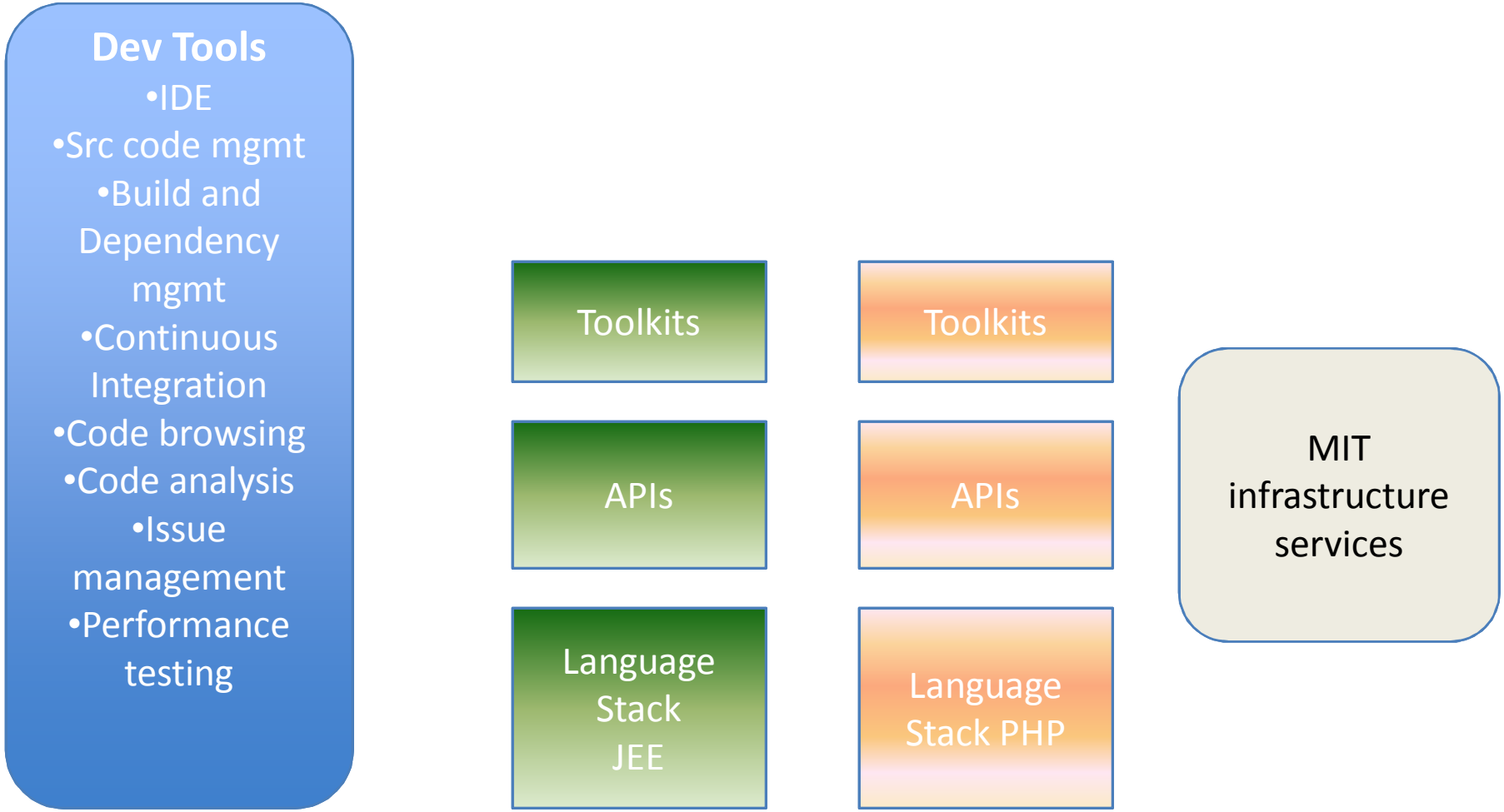
End State: Dev Tools

- Many MIT teams working in a comparable way re: source code management, continuous integration, best practices, testing

Approach to Execution

- Stacks:
  - Prefer open source with paid support when possible
- APIs:
  - Integrate identity services with JEE, PHP stacks
  - Take Kuali into consideration as it unfolds
- Tools:
  - Use best of breed dev tools; open source when possible, but commercial is acceptable
  - Use them ourselves, make others want them
  - Research, prototype, test, then turn over operation to OIS
- MAP working group, steering committee help set on-going priorities as work develops

Conceptual Architecture



Dependencies/Assumptions

- JEE remains the major development stack, growing use of PHP
- SAIS is our biggest customer
- Kuali will be driving Student Vision, and Student Vision will be driving a lot of new software development
- Tools will keep evolving, we will never be “done”
- MAP Working Group, Steering Committee is our governance structure

Risks of Not Doing

- Individual development projects cost more, take longer, re-invent the wheel over and over
- Standards are not adopted, little re-use or compatibility

Risks of Doing

- It takes longer to build infrastructure, vs. just “go do it” on projects
- Wasted effort because developers don't use it
- Partners in the community must build these tools and make sure they work, will other bosses provide the resources?
- Developers like to argue about tools and techniques; not always easy to get agreement
- Standards are immature and always changing, need to develop an iterative approach to providing these tools, which takes resources

Benefits restatement

- Who doesn’t want better, cheaper, faster?