

Buckminster

INTRODUCTION TO BUCKMINSTER

Eclipse Live Webinar, February 2008
Thomas Hallgren (Eclipse/Buckminster)
Stefan Daume (Eclipse/Buckminster)



AGENDA



1. What problem does Buckminster address ?
2. What solution does Buckminster provide ?
3. How does it work ?
4. *DEMO*
5. Usecases
6. Questions

- Component cloud fragmentation
 - Increasing numbers and alternatives to choose from
 - Variations with regard to type, packaging, versioning
- Development fragmentation
 - Variation across repositories (i.e. SVN, CVS, etc)
 - Variation across build & component management systems (i.e. PDE, Ant, Maven, etc)
- No „best solution“ or “one fit for all“
 - Component developers need to be able to speak to all parts of “the Cloud“
 - Component developers need a transparent path through the Cloud and need to get hold of components easily
 - Component developers need to make sure their components can be consumed easily

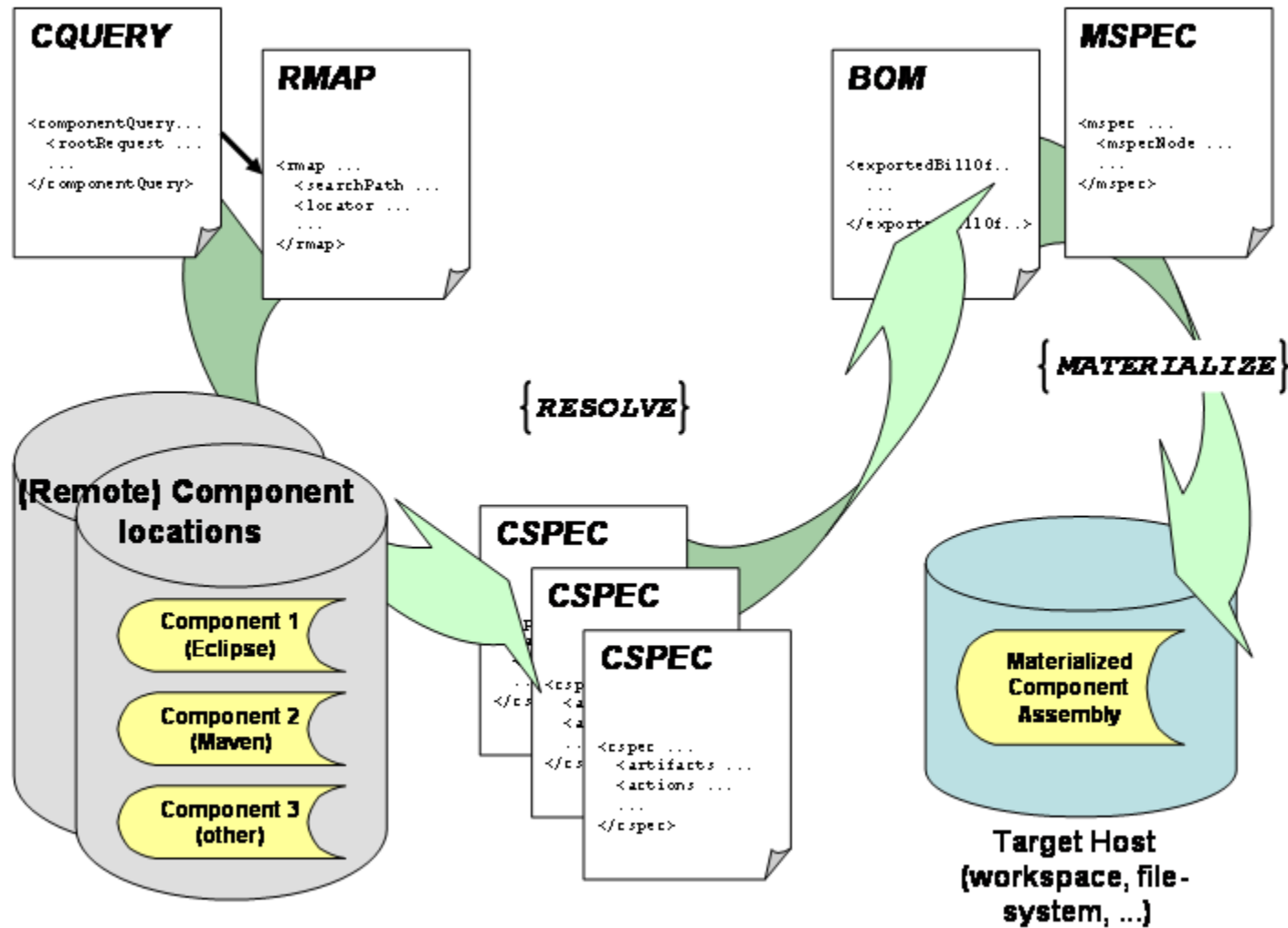


- New member joins development team to work on project X
 - Your colleague says "OK, just check out com.megacorp.overland from CVS and get started".
 - You checkout com.megacorp.overland and discover that it has unresolved project dependencies.
 - "Oh yeah," he says, "you have to checkout project overwater as well." Of course that doesn't completely solve the problem ...
 - "Also you have to get the latest undersea.jar from the central FTP server."
 - "But hey, it doesn't build ok!"
 - "Oh, I forgot, you have to import this jar here, and rename it ... and let me see what else ..."
 - Eventually, after a few rounds of this, you have the whole set of resources and you are ready to go to work.

- Query the cloud
 - Buckminster reduces your quest for a component to a formal query
 - By providing a model agnostic way to talk to the Cloud
 - And it provides sophisticated ways to express YOUR constraints
- A component is ...
 - a name for a thing, be it physical or virtual
 - varying along multiple dimensions (version, mirrors, packaging, etc)
- Buckminster allows you to state something like:
 - "I want version 1.1.0 or a later versions of "that" component in source form"
 - and I don't want to have to worry about dependencies,
 - and I want everything I need to be put into my Eclipse workspace.

- Resolution framework
 - Interpretes and resolves your component query
 - Automatically and recursively obtains component and dependency meta-data
 - Results in a concrete list (“bill of materials”) of locations and actions to satisfy the query
- Materialization framework
 - Interpretes the resolution result
 - And materializes (download, build, copy) all required component artifacts to a context of choice (such as an Eclipse workspace)

BUCKMINSTER – Buckminster artifacts and workflow





- Materialize an existing component
 - ... and those on which it depends
- Publish an existing component
 - ... so that others can materialize your component
- Publish your Eclipse project
 - ... Buckminsterize your project and share it with your team
- Create a virtual distribution
 - ... for a set of various packaged components



- Ganymatic = Ganymede "releng"
 - Current: assembles update sites & distros, manages dependencies
 - Soon: runs tests, verifies community rules
 - Future: measures performance, generates New & Noteworthy, source bundles, etc.
- Built on Buckminster
 - Dependency management
 - Site assembly
 - Runs on build server or on a local developer machine

THANK YOU

visit us at:

www.eclipse.org/buckminster