

## IRC Log Design Discussion 26 May 2005

<brett> profiles: basically a subset of the model is available to create a profile in the pom, which is applied over the other data like inheritance when that profile is switched on

<brett> that serves as a replacement for ./build.properties

<jdcasey> is that going to be overridden by actual inheritance, or what?

<brett> it comes last

<jdcasey> so it'll override even locally specified info?

<brett> it is intended to be local

<trygvis> cli options should come last

<jdcasey> okay

<brett> right, cli always wins

<brett> this comes last in the project assembly

<brett> what did you guys think of the way of doing this?

<trygvis> looks good, seems to solve our use cases

<jason> you mean the ordering?

<jdcasey> well, one point of confusion for me: can we also specify these profiles in settings.xml or somesuch?

<jdcasey> jason: yes

<jason> POM, profile, cli

<jason> yah, that sounds good

<brett> jdcasey: check "location of profiles"

<brett> a subset are also available in settings.xml

<jdcasey> if we can specify a profile outside of the POM, and that profile declares dependencies, haven't we just broken portability?

<jdcasey> yeah, read that

<brett>       \* repositories (which are added to the list in the POM)

<brett>       \* plugin repositories (As above)

<brett>       \* general configuration

properties

<brett> that's all that goes in settings

<jdcasey> ok

<jdcasey> just checking

<brett> that would knock off some extra bugs that cropped up, like the chicken and egg problem with the repositories

<jdcasey> it might also be worth adding some higher level validation like "you shouldn't specify per-user profiles in the POM"

<jdcasey> sure

<jason> did someone post something to the list giving the notice again of our discussion if anyone wanted to join in?

<jason> i'm sure everyone forgets these things

<brett> I will now

<trygvis> I guess the currently used

profiles should be put into the POM when installed and deployed?

<brett> yes, for the project related ones

<brett> jdcasey: I put that down to best practice

<jdcasey> alright

<brett> you may want to check in your own settings - they are identified as being yours

<jdcasey> so we'll detect some profiles' activation based on sysprops, right?

<jdcasey> \${user.name}, etc.?

<trygvis> now, what happens if I build a c++ project targeting windows (profile=windows) and the artifact is deployed

<brett> jdcasey: yep

<brett> trygvis: that is one of my main concerns

<trygvis> I guess the question isn't totally relevant

<brett> you shouldn't be doing it

<brett> but this kind of allows it

<brett> like filtering

<brett> but I couldn't think of a way to satisfy the reqs without allowing that much room for abuse :)

<trygvis> why shouldn't I? it would probably select another compiler (ms's compiler instead of gcc)

<jdcasey> seems plausible

<brett> oh, right

<brett> sorry I misinterpreted

<brett> yes, so you are rebuilding the same project for a different output - so the profile needs to modify the artifact's name, probably classifier

<jdcasey> some of this will require a lot of work on the part of the plugin to respond to profiles like that...

<brett> no, that's the point

<brett> the plugin does what its told

<brett> the profile allows you to configure multiple plugins independently

<jdcasey> yeah, except it'll have to know target-os=windows, and respond appropriately...won't it? oh, I guess it's just plugin config. nm

<brett> right

<jdcasey> brain's still warming up...it'll get better, I promise :)

<brett> :)

<brett> I think this is pretty extensible, like the lifecycle stuff - so even if we don't know how it'll be used in some things like that, it should allow things and we can make it easier later

<trygvis> yeah, the documents are pretty complex. hard to grasp

<brett> they are probably a bit train of thought

<brett> as for the "Bad practices", I think release tools and so on will encourage

people in the right direction moreso  
<jason> and been brewing in brett's head  
since the whiteboard last summer in guelph  
:-)

<brett> :)

<brett> yeah, I got a little excited about  
m2 after that

<jdcasey> I have to say, one other thing  
that I'm a little uneasy about is the  
profile-id naming convention...

<brett> how so?

<trygvis> like the jdk, would be natural to  
have a version range there

<brett> yah

<jdcasey> well, rather than having an id of  
"jdk-1.4" why not split it into to trigger  
elements, something like

```
<part1>jdk</part1><classifier>1.4</classifie  
> (bad names)
```

<jdcasey> then you could just pull \${part1}  
from sysprops to try to match for activation

<jdcasey> if System.getProperty(\${part1}) ==  
\${classifier} then activate

<brett> what if the activation rules were in  
the profile?

<brett> id = jdks-i-like

<brett> <with-jdk>1.4+</with-jdk>

<jdcasey> the only thing about that is now  
you have a whole conditional language you  
have to develop/support, don't you?

<jdcasey> you could make is EL-like, I

suppose

<brett> I can't think of anything outside the jdk, so I didn't want to overcomplicate it for that

<jason> are we going to map an id to some code that will deal with activation of a profile?

<jdcasey> \${java.version} >= 1.4

<brett> down, marmalade boy :)

<jdcasey> hehehe

<trygvis> haha

<brett> hey, why don't we turn the pom into a marmalade script?

<brett> :)

<jdcasey> rofl

<brett> loggy, strike that from the record please

<brett> ok, jason: yes I think so

<brett> I considered that an optional feature though, to start with

<jason> !logging tell john there will be no marmalade POMs!

<brett> user triggered profiles

<jdcasey> brett: what about multiple profile activation from the cli?

<jdcasey> --profiles=one,two,three

<jason> brett: if so then why not just map an id to a class to handled the activation and subclass for things like the jdk

<jason> instead of jdk + classifier

<jdcasey> or multiple --profile= clauses

<brett> jdcasey: definitely, they are cumulative

<brett> jason: I think that's a good idea. so should I strike the standard names, and add some activation triggers?

<jason> i think configurable where we just map ids to the activators and then we can expand as necessary

<brett> yep, ok

<jdcasey> id-pattern -> activators, you mean?

<jdcasey> cool

<jason> the names will be our standards but we can change them at will, or users could change them at will for special cases we won't think about

<jason> jdcasey, yes

<jdcasey> brett: what about collisions on specified configs, etc in multiple active profiles?

<evenisse> how do you define the default profile to use?

<brett> evenisse: the pom itself is the base, there is meant to be a

<id>default</id> as well

<jdcasey> evenisse: should be something like <activeProfiles/> in settings.xml, or sysprops, or on CLI...I think

<brett> the default contains things you don't want in other profiles, but do want by default, if that makes sense

<jdcasey> hmm

<brett> oh, right

<brett> jdcasey: you are right too

<brett> I think that's what evenisse was asking, I answered a different q

<evenisse> yes

<brett> jdcasey: on collisions, I'd just go by the ordering given

<jason> can i try and describe use case, it was going to use activemq but maybe vincent would like to work through a use case with us for cargo

<jdcasey> so can we order the activations?

<jdcasey> oh, right, nm

<brett> jason: go for amq

<jason> ok

<brett> I used cargo to start with, so should look from a different perspective

<jason> the are essentially the same, which is the availability of certain resources

<jason> in activemq they may have a series of tests that execute for every available database available

<jason> the project would define profiles for tests for each of the databases

<jason> now those would be the prototypes

<jason> and provided by the project

<jason> so

<trygvis> sounds like something their IT infrastructure should solve, not maven

<jason> 1) how would the user specify

parameters to plug into the project provided profiles -> say like username/password for the dbs

<jason> and

<jason> 2) would be rely on the user to activate them or could we detect what resources are available to turn on the profile

<jason> say if i had mysql and postgres installed that those tests would run

<jason> in cargos case it might depend on what containers are installed

<brett> yeah, that's what I was going to ask

<brett> if Maven had to re-execute the project

<jdcasey> jason: are you talking about running for multiple db's in one go?

<jason> the whole set of prototypes are available, so do we make the user turn on the profiles for their setup?

<vmassol> sorry guys, I saw you mentioned cargo but the speed is too fast for me to follow as I'm working in the background and watching Rolland-Garros at the same time!

<jason> jdcasey, i'm thinking that on a dedicated integration machine with everything installed the tests would run in one go

<brett> LOL

<brett> jason: inside a test plugin?

<jason> i've been talking with strachan on

how to setup prototype tests for amq using surefire so just trying to figure out how this might work

<brett> so the plugin would take care of it based on its configuration, or would you want the goal to run over and over with different configurations?

<jason> yah, i'm trying to make something for him where we have a prototype test and runs with a different configuration each time

<trygvis> I really don't think that is something that maven itself should handle at that level. what about just configuring the surefire plugin?

<jdcasey> well, each profile could define a goal for surefire:test, and these could accumulate...that would allow them to run side-by-side...

<brett> ok, that sounds best

<brett> jdcasey: I haven't though about accumulating goal definitions, actually

<brett> I think I assumed they'd go over the top

<jason> jdcasey, that sounds on the right track

<jdcasey> but multiple specs of the same goal is something we talked about for the POM right?

<jason> defining configs for testing in the profile

<brett> yes , we had that modello use case for

<jason> i haven't thought it through i'm just presenting the idea

<jdcasey> sure

<brett> ok, let's take it away to think about

<jason> hokay

<jdcasey> it's getting back to that same old problem of when to accumulate, when to override

<jdcasey> k

<jason> so basically on the integration machine you run everything, in another environment you run what you can

<brett> right

<brett> jdcasey: yes, we need to keep it clearly defined when each happens

<brett> I think here, multiple profiles would add new goal calls, plugin management used to apply settings

<brett> and we never override

<brett> consistent with inheritance

<jason> so a profile provides a clean slate for execution?

<jdcasey> that sounds good to me

<jason> that would be clean

<brett> jason: yes, but it doesn't cause any additional executions, it just augments the one project

<brett> ok, so shall we move on to the next,

or are there more issues?

<trygvis> hang on

<trygvis> I was thinking about these profiles in relation to Continuum

<brett> ok

<trygvis> just got to recover my thoughts

<jason> the building of the runtime?

<jason> dev vs production?

<trygvis> no, for continuum to know what JDK to use

<jdcasey> continuum should provide alternate activators, IMO...to allow simulation of any build environment, not just the one on the integration server

<trygvis> possibly dev vs prod too

<jason> i think that's a runtime/app thing

<trygvis> jdcasey: it should be user configurable

<jason> ok, i think those are continuum app specific things and not really anything to do with profiles per se

<jdcasey> trygvis: yes, but maybe not at the continuum-instance level...maybe more at a per-project level, shouldn't it?

<trygvis> it would also be useful to use continuum to try and build a project in another enviroment, like for checking with jdk5 compatability

<trygvis> jdcasey: yes, per project configurable

<jason> ok, i think thats' out of scope for

this discussion

<trygvis> ok

<brett> alright, next one is a bit lighter  
<trygvis> just wanted to make sure continuum  
can make use of this information

<brett>

<http://docs.codehaus.org/display/MAVEN/Custom+Repository+or+Resolver+Implementations>

<brett> this is more something for me to  
understand

<brett> should/how can users have custom  
implementations of components

<jason> i would say at the transport level  
yes

<jason> like implementing new wagons

<brett> yes, that's what I was hoping

<jason> yah, i think that's fine

<trygvis> hmmm

<brett> but if continuum was going to do a  
whole snapshot build

<brett> would it want to override the  
transformations?

<jason> the layout and resolution mechanism  
i would say no

<jason> brett, these transformations we are  
starting to use as maven specific things

<jason> i would rather that nomenclature be  
standard

<jason> like that guy wanting to use their own custom stuff?

<jason> i think that's a stretch for a general use case

<brett> yeah, I listed that as a nice to have, for sure

<trygvis> I think the linux distros would like maven (also maven component use elsewhere that in m2) to use their layout <trygvis> which basically is the same issue as that guy had

<brett> that was the main motivation here

<jason> the layout really wouldn't be technically hard because requests for artifacts are no longer path dependent but do we even want to do that

<jason> what kind of layout do they want?

<brett> usr/src/.../lib/lib name

<brett> hi Arnaud

<Arnaud> hi

<brett> ok, so how about we prefer to just allow different wagon implementations, and they construct the metadata from their own

<jason> sorry, did anyone say anything about the distro situation?

<brett> not much

<jason> wagon implementations for sure

<brett> we can work with them to see what we can do, but its not going to seriously impact us right now

<jason> maven artifact allows different

layouts but do we even want that for maven  
<trygvis> even if we can have debian-wagon  
we still need to map the artifact to their  
path

<Arnaud> I'm at work thus I'll be discreet  
;-)

<jdcasey> jason: we were just saying that  
the whole repository concept's a bit  
overblown, and we should just go back to  
\${project}/lib/\*.jar :)

<jason> my gut is that it would become a  
problem because people would just implement  
different layouts because they can and it  
would be arbitrary

<brett> ok, will strike that

<jason> for maven itself does anyone think  
we need different layouts?

<trygvis> no

<brett> no

<brett> only for backwards compat

<jdcasey> no, but I think it's a useful  
feature to have in case we find a better  
layout later on.

<jason> people can do what they want with  
maven-artifact but for maven use i don't  
think so

<jason> jdcasey, sure

<brett> jdcasey: exactly

<evenisse> no

<jason> our modifications fine, arbitrary "i  
want my own layout because my company blah

blah blah ..." definitely not

<brett> ok, so people dropping in alternate resolvers and such basically need to roll their own maven?

<jdcasey> and as far as the custom resolver goes, IMO we can't risk dropping or poorly supporting the transformations for this...

<jason> i think how we deal with snapshots and releases is core to maven and i think wiggling around that will just get people in trouble

<jdcasey> +1

<brett> ok

<brett> we can work with the distro people to see what they want and push back, but it shouldn't affect the architecture at all

<jason> +1

<trygvis> I think that is possible given the low number of java packages in the distros

<brett> ok, anyone up for dependency mediation?

<jdcasey> bring it on

<brett> jdcasey: you said you had some feedback on that?

<jdcasey> it's about the whole resolve-to-concrete thing on release

<brett> yep, that's important.

<brett> I had a couple of questions myself in the yellow box there ;)

<brett> what's your Q?

<jdcasey> what about creating either (a) a special release-POM with all deps resolved concretely, or (b) a special tagged version of the POM with that info, followed by another non-tagged commit of the user-defined POM

<jdcasey> this is something it seems that you're wrestling with in that document...

<jdcasey> I'm -1 on losing user info in favor of resolved deps...but that's an obvious statement

<brett> yeah, I didn't think of recreating the pom with resolved stuff, I was just going to mark it up

<brett> which also facilitated doing it in

SCM

<jdcasey> mark it up how?

<brett> that extra resolvedVersion tag

<brett> maybe it could add a profile :)

<jdcasey> that's an idea, but it'll lead to some serious bloat, and for something that's essentially a snapshot in time

<jdcasey> IMO so would the resolvedVersion stuff

<jdcasey> resolvedVersion would get stale pretty quick, and probably would be removed before starting development of the next release anyway

<brett> definitely

<jason> what is scope=none? only resolved for the project?

<jason> the example on the top there

<jdcasey> but wow, this represents a big shift in maven philosophy wrt versioning...now we're starting to enforce a versioning scheme of sorts

<brett> jason: that was how I figured to exclude transitive deps you don't want

<jason> when would you want to do that?

<jason> is the jdbc thing a real example?

<brett> ideal world: never

<brett> but we can't leave projects at the mercy of their dependencies

<brett> so this would be how they get to have final say on their deps by overruling

<brett> rather than turning of transitivity

altogether

<brett> which would just be abused  
<jdcasey> but if a dependency declares that  
it needs such-and-such, can we really  
second-guess that? it seems like it would  
lead to bugs/errors

<jason> did someone have a case where  
scope=none helped?

<brett> jason: you'd use that example on  
velocity, for example

<trygvis> the continuum plexus app

<brett> when you have a dep on velocity, and  
it gives you jdbc

<trygvis> and logkit+avalon framework api

<brett> jdcasey: we're not second guessing  
as such, its just making sure the project  
gets to have the final say

<brett> I think that's important, both on  
inclusion and version

<jason> yah, that's a crappy build

<brett> right, and there will -always- be  
crappy builds

<jdcasey> so if I define a dep with  
scope=none, it'll just blow it out of the  
deps list wherever it encounters it?

<jason> nooooooooo

<brett> jdcasey: right

<brett> jason: I know, but...

<jason> i think some extra deps there might  
be better then scope=none

<jason> might it lead to some problem, where

you did that and have another dep that needs jdbc

<jdcasey> exactly

<brett> yes, it requires some intelligent use

<jason> just playing devil's advocate and what's better, some extra deps or something that potentially doesn't work

<brett> not when they get bundled

<brett> not when they don't exist in the repo

<jason> sorry, not following

<brett> not if they are not licensed appropriately (when we start checking that)

<brett> examples when it is not better to just get the deps

<jason> what do you mean "not when they get bundled"?

<brett> way I see it is that the project should always have the final say, but this is somewhat annoying enough that they will try and get the source of the problem fixed

<brett> look at continuum's lib directory

<brett> I know they could be excluded in other ways

<brett> but that's the sort of thing we're trying to avoid

<jdcasey> brett: how will the scope=None stuff work when it's in a project that's brought in through transitive resolution?

<jason> yah, not ideal but i can live with

that knowing that the velocity build will improve

<evenisse> If I use a dep that define a dep with scope = none, would I obtain it or is it excluded only for the project?

<brett> jason: you can. other's won't

<jason> i think the scope=none could be potentially cause problems

<brett> ok, alternatives to solve the same problem?

<brett> or is it solving the problem that causes problems?

<trygvis> a better application for managing the repository so stuff like this can be fixed in a better and more permanent way

<jdcasey> trygvis: the avg. user won't have access to such

<jason> i don't think getting some extra bits is a huge problem

<brett> ok, jason volunteers to answer that question on the user list every day ;)

<jason> i will get loggy to do it!

<jdcasey> brett: I think one thing that would help immensely is knowing what implies what...seeing the dependency graph, in other words

<trygvis> we'll just use bayesian filtering on the user list to point people to the FAQ

<trygvis> like we talked about jason

<brett> jdcasey: I agree

<jdcasey> a tool to do that, plus a

local-only way to "fix" poms might do the trick

<brett> local fixes is problematic

<jason> repoman

<jdcasey> b/c of portability?

<brett> yep

<jdcasey> agreed

<trygvis> because it's local :)

<brett> rm -rf local\_repo breaks stuff

<brett> repoman is better, but we really want to hear about it

<brett> everyone using scope=none emails the maven developers automatically :)

<jdcasey> you mean setup a trigger in maven to actually notify the list (or whatever) that this pom is broken? :)

<brett> ok, lets put it aside as problematic. I think we need a solution. I don't think it is acceptable to just drop in extra deps for the reasons given above.

We'll come back after more thought

<jdcasey> brett: [repost] how does scope=none factor into poms resolved transitively?

<jason> it just gets taken out of the graph entirely

<jdcasey> ignored (I hope)?

<brett> not sure

<jdcasey> I mean resolved poms with a scope=none in it

<jdcasey> ok, just file it away then :)

<brett> depends where depMgmt is applied on transitive poms

<brett> will also be affected by versioning, so I'll just note it

<brett> I think "changes to dep mgmt" actually describes how it works now

<brett> except for the last point, which is just scope=none

<brett> changes to scope element can be skipped

<brett> everyone agree "resolve dependencies once only" is sane?

<jdcasey> I like that one, actually

<jason> +1

<brett> ok, something more tricky

<brett> version ranges

<ben> oh that'll be fun

<Arnaud> +

<jason> would be good to get one of the SAS guys here

<jason> i'm going to mail one of them, they probably have some ideas wrt versioning

<jason> having 300 builds

<jason> and if they would go for a standard versioning scheme

<brett> will gwaltney one of them? he was here 1/2 hr ago (I assume that was him)

<jdcasey> jstrachan might have a problem with it ;)

<jason> yah, he's one and jared

<jdcasey> active-\* always causes problems on

conversion with version names

<brett> what did you think of the versioning scheme?

<jdcasey> brett: what's the diff between '1.0' and '(1.0)'?

<brett> nothing

<jdcasey> oh

<jdcasey> okay, then

<brett> "Soft requirement, equivalent to 1.0 on its own

<jdcasey> :)

<brett> but [1.0] would be absolute

<jdcasey> right

<brett> does the syntax suit people?

<brett> very math oriented :)

<brett> kind of ugly though

<jdcasey> nah, syntax is good for me

<jason> i think it's good, being math oriented we know it already works

<jason> set notation good

<jason> no one can argue with us

<brett> I bet someone has a version alpha(2) :)

<ben> not on windows they don't

<brett> ok, we'll stick with that for now

<brett> I like Jason's point too

<ben> oops I tell a lie, ( is valid on windows

<jdcasey> brett: will 1.0-RELEASE and 2.0-RELEASE be ok together?

<brett> no such thing

<jdcasey> you can't be maintaining two release cycles on a product?

<brett> I really prefer RELEASE be silent and behind the scenes

<brett> that's not what it was intended for...

<jdcasey> ok

<brett> we might need to re-evaluate, its mainly to support the plugins right now

<jdcasey> misunderstood the finer points of RELEASE, I guess

<brett> you're right, which is why I want it hidden

<brett> if that makes sense

<jdcasey> I think so

<brett> ok, are we happy with the technique of version comparison, and the version layout?

<jason> +1

<jdcasey> I think those types of notations (like -SNAPSHOT) might be useful eventually, but for now, that's no biggie

<jdcasey> +1 here

<brett> shall we change our snapshots to have \_build then?

<jason> i pinged walt to see if he wanted to show up

<brett> cool, we'll skip over the next bit until then as that's the important bit

<jdcasey> brett: I think so

<jason> as long as you stipulate deployment

to one repository or else you won't be able to increment accurately

<jason> which we are doing now yes?

<brett> right

<jason> then that's cool

<brett> except under profiles, but that's intentional

<jdcasey> brett: we may have to move the artifact classifier too, just to make the whole thing more parseable

<brett> ie, you are incrementing different builds

<brett> jdcasey: not intending to have to parse it in the repo

<brett> you just need group ID and artifact ID

<jdcasey> actually, strike that anyways

<jdcasey> right

<brett> that identifies the project, and that has all the info

<jdcasey> I meant for the repo mgt tools

<brett> yep, it can find the project.

<brett> next point was "preventing RELEASE dependencies", which I explained earlier.

Let them specify an open ended version instead

<brett> so [2.0) is 2.0-RELEASE

<jdcasey> is resolvedVersion on hold pending further thought?

<brett> yeah, will do that last if SAS guys turn up

<jdcasey> k

<brett> hmm, did I screw up set notation?

<brett> I think ) actually means "don't include this"

<jason> look on mathworld

<jdcasey> should've known it would be a wolfram site

<brett> ok, I'll check later

<jason> yup

<brett> lot of good that maths degree did me

<brett> agree on parent versions?

<brett> "They do not need to incorporate ranges, and are generally treated as unversioned."

<jdcasey> brett: I suspect you're right on being wrong, tho... ;)

<jdcasey> what do you mean by unversioned?

<jdcasey> they can change, no?

<brett> yes, to the user they generally want the latest

<brett> but it gets resolved on deploy

<brett> they can specify it, but generally they don't want to

<brett> actually, that's part of the release mgmt discussion, and JASON and I spent a fair while on it before, so maybe we should come back to it

<jdcasey> ok

<brett> nearly to the end of this document, then might be time for another break :)

<brett> agree that version ranges wouldn't

match a snapshot unless it is specified as a boundary?

<jdcasey> +1

<evenisse> yep

<jason> +1

<brett> ok, so all that is left is conflict resolution, and reproducibility

<jdcasey> I'm interested in what would create a warning message as being potentially workable, but outside the specified range(s)

<jdcasey> is that just any dep that is specified by group/artifactID, but where the version is incompatible?

<brett> that may be redundant, I had it written earlier

<brett> where is it again?

<jdcasey> h/o

<jdcasey> maybe I made it up...can't find it

<jdcasey> :)

<jdcasey> disregard it, then

<brett> no, I said it

<brett> oh, its just in the original reqs

<brett> they are not part of the design, just the inspiration for it

<jdcasey> well, that would apply in "use-nearest" after a fashion

<brett> yeah, I think using soft requirements solved that

<jdcasey> if you use the nearest, and another transitive dep had that artifact

with a different version spec, you might want to drop a warning

<jdcasey> ok, cool

<brett> right

<jdcasey> where is that strategy best defined?

<jdcasey> I might argue for the settings.xml, personally

<brett> needs to be consistent for different people building the project

<jdcasey> per-project might leave it open to changes injected by POMS of transitive deps, though...

<jdcasey> I agree on that

<brett> I don't think you'd honour what a transitive pom did

<jdcasey> guess you could specify to use the strategy in place, and ignore those specified by POMS of transitive deps

<jdcasey> ok

<brett> I'd like to start with just the one strategy so we get a feel for how it works, and see whether others are really needed

<brett> easier to give than to take away

<jdcasey> so start with the range-based?

<brett> yep

<jdcasey> +1

<jason> just a note, if we're going to start using a graph proper then let's make a separate project of that tool because i want to be able to visually represent the graph

with decorations on the nodes for all the items we're talking about

<jason> +1

<brett> agreed

<jason> i want to use the exact same tool for visualization

<jason> jung.sf.net is very cool btw

<jason> i've settle on thta

<brett> ok, I'll take a look

<brett> do we think we'll take a crack at the reproducibility question?

<jdcasey> I think we have to, if we're going to get into compatibility ranges at all

<brett> yeah, I just mean now :)

<jdcasey> we have to have a concrete, reproducible build for release

<jdcasey> ah

<jason> if we're doing ranges are we implicitly going to promote our versioning strategy

<brett> I think we only support our versioning strategy for ranges right now

<jason> i would like

<jason> i think if we have something reasonable, people will use it

<ben> could you not emit a file containing the decisions the automater took

<ben> then use that as an optional input at another point in the future

<jdcasey> ben: like a properties file or something of resolved versions, as a

separate file?

<brett> jason: it encompasses most of the standards now, although changing build to \_ differs from RPM

<jdcasey> the only problem with that might be portability

<brett> what do you mean jdcasey?

<jdcasey> was talking to ben :)

<brett> I know

<jdcasey> I mean the resolved-versions file would have to be deployed too...wouldn't it?

<jdcasey> unless we have a fully-resolved pom that we release to the repo

<jdcasey> that means a pom with all transitive deps included and pegged, no?

<brett> ok some objectives:

<jdcasey> actually that might be a really good thing to do...flatten it all out for a release

<brett> - capture all of the information used in a release

<brett> - retain the user's original specifications

<brett> - store at release point in SCM, and repo

<brett> the question I have is, when do we use the resolved versions, and when don't we?

<brett> hi jared, willgwal

<brett> brb, grabbing coffee. feel free to discuss

<jdcasey> I'd say that you're going to have a problem any time you're unable to depend on a release-quality POM...you don't know how it'll resolve those transitive deps

<brett> that's what's getting to me

<jdcasey> I think that on release you should build a release-quality POM that has all transitive deps included in it, with concrete versions for each. This POM shouldn't replace the pom.xml for the project, but should probably be put into version control for future reference

<brett> if you always take what the release used, you are losing the ability to make a smarter decision knowing all the factors it had available

<brett> if you always take the other factors, you might get a different result

<brett> jdcasey: sure, that's for building that project. I agree 100% on that

<jdcasey> but do you want to make a different decision? that's not reproducing the build, and could mean that debug/test builds fail to reproduce a bug

<Arnaud> +1 for jdcasey and POMs

<brett> here's another point: shouldn't it always make the same decision?

<brett> the released poms don't change

<jdcasey> aren't you assuming that all the transitive deps will be releases?

<brett> for a reproducible build, yes

<brett> the release plugin should just freak out at snapshots

<jdcasey> but as a user you don't always have that kind of control

<jason> releases or just non snapshots?

<jdcasey> jason: what's the difference?

<brett> is there a difference?

<brett> hehe

<ben> but lets say commons-foo depends on commons-clown [1.\*]

<ben> and commons-clown 1.1 is out for your original build, but after than, commons-clown 1.2 came out

<evenisse> I'm sorry, but I must leave.

<jdcasey> what about the timestamped deploys that go with snapshots? can't you depend on those for a release?

<jason> jdcasey, that's what i mean

<jdcasey> yeah, just got that :)

<jason> it doesn't necessary have to be a proper release, just a timestamp

<jason> ok

<jason> sorry, i have twins running around at my feet

<jdcasey> that's why we can't depend on all transitive deps to supply release-quality poms

<ben> jason: I have a newfound appreciation for the distraction that children cause :)

<jason> oh yah

<jason> that's why i borrow them :-)

<brett> LOL

<jason> i'm enough of a scatter brain

<brett> the problem with timestamped snapshots (at least now) is that the deployed poms may contain unresolved snapshots

<jdcasey> right, which means that when your project does a release, it has to resolve and then peg those snapshots for its own purposes

<jdcasey> so it has to collapse the transitive deps into its own pom

<jdcasey> it'd be nice to annotate them as to their origin

<jason> what if decided that anything deployed had to be resolved?

<brett> hold up

<brett> 3 questions I can answer here

<brett> actually, 2 I forgot what I had to say about clowns

<trygvis> they rock?

<brett> folding down transitivity:

<brett> this means you can't effectively use the ranges any more

<brett> unless you retain the spec to use later

<jdcasey> hmm

<jdcasey> I see what you mean

<brett> [1.1) might resolve to the latest 2.0, but your project wants [1.2] exactly, no way to find out what was really wanted

<brett> jason: resolving snapshots on deploy means you don't get updates any more, basically same reason

<brett> you lose info

<brett> is that right?

<trygvis> this is something that all the linux distros have fixed, we should perhaps look more into what they've done

<jdcasey> maybe we need a built-with metadata file, then...more of an audit than anything else

<jason> what are you losing if the POM is fully resolved for deploy?

<brett> trygvis: have they?

<jdcasey> something that we could use as input

<jdcasey> jason: you lose the compatibility statement provided by that range

<jdcasey> for other projects to use

<brett> s/range/snapshot/

<jdcasey> range on the dep, I mean

<jdcasey> or snapshot, yeah

<brett> a SNAPSHOT depends on b SNAPSHOT (1), c depends on a SNAPSHOT and hence b

<brett> a gets published, is now (1) with dep on b (1)

<brett> new b (2) is published

<brett> c ->a (1) -> b(1)

<brett> but you want b(2)

<jason> yah but for a release how are you ending up with a snapshot depending on a

snapshot?

<jdcasey> brett: I'd agree, with one thing: major version changes are meant to signal API incompatibility, no?

<jason> oh, the ranges

<jason> jdcasey, typically

<jdcasey> so maybe [1.0) would mean anything in the 1.x range, but not 2.0

<brett> I don't think so

<brett> they might be backwards compat

<jdcasey> [1.0,2.0) would be correct for that case, then?

<brett> yes

<brett> interesting thought there

<brett> hard to retrofit onto something that used to declare [1.0)

<trygvis> on debian they would change the artifact id if they change stuff (radically)

<brett> typically a good practice, I think

<jdcasey> so it'd go from foo-1 to foo-2 artifactID?

<brett> but we can't rely on it. People break stuff between 1.1 and 1.2 :)

<trygvis> that's how they do it

<brett> though that was pretty ugly when you showed it before :)

<trygvis> libcommons-collections-java

<trygvis> libcommons-collections3-java

<brett> ok, noting the retrofitting as an issue

<brett> actually, the dependency management

would just be used to override the  
over-enthusiastic pom

<adc> Get an error running the m2 tutorial  
<adc>

<http://rafb.net/paste/results/SMNoHH78.html>

<brett> adc: I forgot to update it I think

<brett> with the last release

<brett> which was dumb of me

<brett> will take a look later

<brett> ok, trygvis has a good point

<brett> I think the packagers take the  
approach of "one version to rule them all"

<trygvis> they do

<brett> and if they don't play together, its  
a different package

<brett> but that has come with its own set  
of problems, too

<jason> yes, that is one thing etrade  
specifically didn't like about gentoo

<jason> and why they didn't use it

<brett> right, we don't have to do that. we  
can have every version known to man around  
for use

<jdcasey> yeah, like try tracking an  
artifact across all the different  
artifactId's

<trygvis>

<http://rafb.net/paste/results/LpedXM40.html>

<brett> trygvis: your point?

<trygvis> just showing how they specify deps

<brett> k

<brett> I'd like to use  $\geq$  syntax, but not in XML :)

<trygvis> kaffe ( $\geq$  2:1.1.4.PRECVS8-2)

<jdcasey> I think we need to get the version-range syntax nailed down, but I think this stuff will work.

<trygvis> they also have some issues with they version stating although they're making it more flexible

<brett> the  $|$  is interesting, kind of like specification deps. I've ruled them out for this release though

<jdcasey> (having a syntax makes it a little easier to explore it all)

<trygvis> I like what brett proposed, yay mathematical synatx

<brett> as long as I get it right

<jdcasey> and we all learn it ;)

<jdcasey> I didn't have set theory in college

<trygvis> <http://en.wikipedia.org/wiki/Set>

<trygvis> ;)

<brett> I did a bit of everything. A lot of pure maths is good for computer science, if useless in the real world :)

<jdcasey> yeah, been reading that in spare moments, trygvis ;)

<trygvis> I like the power set character

<trygvis> looks like a super fancy P

<jason> brett: that's not true. pure math is very handy

<jason> you wait till we play graph  
visualization :-)

<brett> yeah, it was tongue in cheek, mostly

<brett> ok, let me summarise where we are  
at...

<brett> I think we've agreed on everything  
in that doc

<brett> but with the following issues

<brett> - strategies are out for release 1,  
adding complexity to a problem not fully  
tested yet. likewise spec deps

<brett> we'll just use the range

<adc> brett, ping me when you're ready to  
help me out, tks.

<brett> adc: will do

<brett> - we need to get to the bottom of  
what gets resolved, how, and when it is used

<brett> did we all agree that resolving  
versions in the transitive poms outright  
would limit the ability to do the right  
selection later?

<jdcasey> yes

<brett> I'm thinking we operate in two  
modes, basically. use resolved versions  
(historical builds) and used everything

<brett> ok, I think we can keep this simple

<brett> I think we can ensure the  
reproducible build is self contained

<brett> that eliminates all of the guess  
work

<brett> I don't think that needs to go into

the repo at all, only SCM/source distro

<brett> WDYT?

<jdcasey> might also put it in the jar itself, for reference

<brett> this is essentially what ben said at the start

<jdcasey> sort of a "here's how I was built" thing

<brett> jdcasey: right. We actually agreed to do that already last week, didn't get around to it. This takes it a step further.

<jdcasey> k

<jason> so effectively a manifest for reproduction?

<brett> right. if you are using a source build, transitivity is turned off, and all the versions are resolved

<brett> not sure about the parent

<jason> so that pom would enumerate the entire list of artifacts required to build?

<brett> right

<brett> makes it immune from repo changes, changes in the app code, etc

<jdcasey> s/required/used/

<brett> jdcasey: good point :)

<brett> I can't see a use case for deploying fully populated poms to the repo - we went through that the other day and we agreed it was a bad idea

<jason> really with ranges there is no other way

<brett> and meanwhile, normal source code builds will always use the intelligent version

<brett> its nicely paralleled to the snapshots, so I like it

<jason> so for a source drop that POM could be placed in there so it would always work, what would we do with the SCM as far as the build being reproducible?

<brett> drop it in, tag, revert back and bump version.

<brett> no more commits than we have now

<brett> bigger diffs :)

<brett> I'm open to an alternate way of doing that though

<jason> so fully resolved pom gets checked in and revert back again

<brett> maybe a separate file

<jason> ok

<brett> so, if we agree on that, outstanding issues are:

<jason> so tag with the full resolved POM and then the perform will build using that fully resolved POM?

<brett> correct

<jason> that makes sense

<brett> ok, issues:

<brett> - how do we resolve the parent?

<brett> - how do we store the information? (in the pom, replacement pom, side by side file)

<brett> parent, I guess we also fold in.  
<jason> fold in how?  
<brett> I was thinking we needed to keep it dynamic because somethings can change (eg the company URL), but a release is a release  
<jason> there won't be any ranges there and we could always retrieve it no?  
<brett> fold in, I mean write out the model at the end of assembly, and keep that  
<brett> the parent has deps that would affect the final pom  
<jason> so fully resolve parent and all  
<jdcasey> brett: I'd be +1 for making it totally flat for historical purposes...nothing to lose that way...  
<brett> yep  
<brett> yeah, I think so  
<brett> ok, last decision in this marathon topic... where do we store it  
<brett> I'm in favour of a separate file actually  
<jdcasey> yeah, me too  
<jdcasey> seems cleaner  
<jason> you mean the fully resolved POM?  
<brett> yep  
<jason> the full-monty-pom.xml  
<jdcasey> pomzilla.xml  
<brett> uberpom.xml  
<brett> well, the profiles doc already called for profiles.xml  
<jdcasey> nice side effect is to be able to

diff releases of that file for differences between releases

<jason> so in a source drop or tag it's just pom.xml but where is this uberpom going?

<brett> next to it

<brett> if it exists, use it instead (unless overridden by switch?)

<jason> the same file?

<brett> release-pom.xml

<brett> not sure what you mean?

<jdcasey> no, we'd leave the original pom.xml untouched, except for possibly incrementing the version (?)

<brett> right

<jason> if i just checkout a tag and go "m2 install"

<jdcasey> this is instead of replace-tag-revert, isn't it?

<jason> so the tag will have the original pom.xml and to reproduce the build the release-pom.xml would be used

<jdcasey> that's what I think

<brett> yep, added bonus is that it would remove the need for release.properties

<brett> I think

<jason> ok, but i think everyone will check out the tag and just "m2 install"

<brett> yep

<jason> which won't use what we intend

<brett> hmm

<jason> ok, i'm confused

<jason> the pom.xml in a tag will be full resolved on or not?

<jason> s/on/one/

<brett> I was thinking not, so it didn't have such a wild change history

<jason> that's fine, we could probably have the cli look for the release pom so people don't inadvertently build with the wrong pom

<jason> they will naturally just check out and go "m2 install"

<jason> or at least warn them there is a release pom there that can be used

<jdcasey> sure

<brett> do we want to keep the release pom there, or only have it on the tags?

<jdcasey> I'd think only on the tags

<jdcasey> since it's only valid for that release

<jason> i think only in the tags and source balls

<jdcasey> that file will go stale as soon as new development starts

<brett> ok, so release process is:

<brett> - set release version in pom.xml, write out release-pom.xml

<brett> - tag

<brett> - delete release-pom.xml, bump version in pom.xml

<brett> correct?

<jdcasey> +1

<jason> yes

<jason> why don't we do the last two which are short and then we'll tackle the two bigger ones tomorrow

<jason> plugin report inher and release management

<jason> the lifecycle one will take a while i think

<brett> yah

<brett> ok, trying to get to the plugin/report inheritance doc...

<jason> oh confluence

<brett> refresh...

<brett> there we go!

<brett> so, default goal is inherited

<brett> mojos will become inherited by default

<brett> ie, plugin configuration

<jdcasey> wow, ok

<brett> but a mojo can declared that it desires its default is to not do so

<brett> jdcasey: background. there was a use case where it was desirable

<brett> eg, you have 30 projects using aspectj that have a common root

<jdcasey> hmm. figured someone screamed loudly enough

<jdcasey> sure  
<jdcasey> and pluginManagement didn't solve it adequately...  
<brett> no, because it doesn't bind  
<jdcasey> because you still have to declare the plugin everywhere  
<jdcasey> ok  
<brett> we still have both, and this makes it completely consistent with dependencies  
<jdcasey> ok  
<brett> consistency good  
<jdcasey> sounds fine  
<jdcasey> :)  
<trygvis> there is a issue with the pluginManagement, couldn't get it to work the plexus-components  
<brett> trygvis: file it  
<jdcasey> issue with pluginManagement, or issue with trygvis? :)  
<trygvis> not sure it's a bug  
<trygvis> exactly  
<brett> post it :)  
<trygvis> we can talk about it later  
<jdcasey> trygvis: I can help you with it offline, if you want  
<brett> posting to the list is good, because others who make the same mistake can google it  
<trygvis> or just fix it :)  
<jdcasey> or better, file it, and we can close it if it's operator headspace error

<brett> back to mojos declaring their default inheritance...

<jdcasey> yeah, post to the list

<jdcasey> ok

<brett> so mojos where it doesn't make sense (eg, assembly) can say it defaults to not being inherited

<brett> and can be overridden by

<inherit>true|false</inherit> in the goal or plugin section in the POM

<jdcasey> and that's specified in the parent-pom, right?

<jdcasey> guess it would have to be...

<brett> I was thinking child

<jdcasey> oh

<brett> good point, I think I was

<jdcasey> so basically says "bring in the settings from the parent for this"?

<jdcasey> has to be parent, right? so we can avoid having to declare aspectj all over?

<brett> heh, yeah

<brett> updated doc

<brett> ok, reports behave identically to plugins

<jdcasey> OT: are reports == plugins?

<brett> and lists merge

<jdcasey> k

<brett> yes, they are

<jdcasey> k

<brett> so, following tradition: no way to block out something inherited

<brett> you fix the parent

<jason> coolio

<jdcasey> right on

<brett> ok, anything else of concern there?  
would that simplify the documentation task?

<brett> by that I'm referring to the doc  
Matthew Pryor was writing

<brett> because it was all kind of confusing

<jason> did he put it somewhere?

<jdcasey> I think so...I need to re-read it

<brett> he mailed it to us

<brett> I asked him to put it somewhere,  
haven't heard back

<brett> I still have it, but it needs some  
more fixes

<jdcasey> the consistency aspect should make  
things clearer, if nothing else.

<brett> okie dokie

<brett> releases

<brett> we may have already solved some of  
this, let me re-read

<jdcasey> the default inheritance setting  
for a particular mojo might be a bit  
confusing. it requires a look at docs, but  
the behavior should make it apparent what's  
happening

<brett> jdcasey: right. It could be abused,  
but I figured it should appear to the user  
that it always works as intended - some  
plugins are meant to inherit, those that  
aggregate wouldn't

<jdcasey> sure

<trygvis> I'm off

<brett> later

<jason> later

<brett> ok, the release stuff is pretty straight forward at first

<brett> but the branches, etc still throw me for a loop

<brett> so what we are trying to figure out here is really how versioning works in a multiproject

<brett> and how the release tools can cope with this consistently - but I guess since it is folding in the pom now, its not such a big deal

<jdcasey> it's the pathing for branches in svn that makes this hard, isn't it?

<brett> yeah, but the concept applies to cvs too.

<brett> if cvs works, we can endorse svn switch as the way to do things

<brett> but I'll leave that to last

<jason> actually can we do this one tomorrow?

<jason> i want to think about continuum for a little while in terms of a release

`<brett> okie dokie`

`<jason> that was good though, that went fast`