DeltaStreams

Changesets on steroids

I am...

Göran Krampe, goran@krampe.se

Live north of Stockholm, Sweden

Wife, daughter, cat, boat, garden...

Consultant & group manager at MSC

Smalltalker since 1994 approximately

History
(as memory permits)

- Started in aug 2007
- Significant help by Matthew Fulmer
- Development slow at times

...revived before the summer!

Inspiration
(cool stuff that actually works)

- Monticello
- Changesets
- Darcs
- git

Monticello
(Avi's baby born from an OOPSLA discussion)

- Really nice for cooperative development!

But...

- Needs common history
- Snapshot centered (not changes)
- Package granularity

Changesets
(sneaky beasts...)

- Nice ”patches” by being simple!

But...

- Non intuitive implementation
- Captures too little
- No ordering
- Could be much better...

Darcs
(DSCM magic deluxe)

- True out of order cherry picking!
- True out of order revert!
- Truly change centered
- New ”types” of changes
- Selective commit

Git
(The KING in the real world)

”THAT is what merging is all about. Not smart merges. Stupid merges with good tools to help you do the right thing when the right thing isn't _so_ obvious that you can just leave it to the machine."

Linus Torvalds, 2006

...so Delta
(a ”patch” the Smalltalk way)

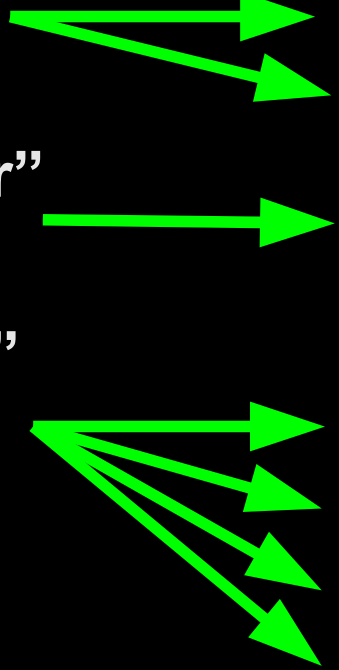
- Simple and change oriented
- Not relying on history
- Good base for selective commits
- Can revert and cherry pick
- No magic!

Design
(my confused brain)

- A Delta is an ordered list of Changes
- Class hierarchy of Changes
- Each change:
    - Captures state ”before” and ”after”
    - Captures enough state to do revert

Pin pointing intent
(capturing more stuff)

Changeset	Delta
”method is this”	”add this method” ”change this method to that”
”remove selector”	”remove this method”
”class definition”	”add ivar x” ”remove ivar x” ”change superclass” etc etc etc



Demo 1
(pretty please, feed us code!)

- Trivial stuff:
  - Create a Delta
  - Record some changes
  - Explore it

Design
(back to boring slides...)

```
Object #()
        DSAnnotatedObject #('timeStamp' 'properties')
                DSChange #('number')
                        DSBasicClassChange #('className')
                                DSClassChange #('superclassName' 'instVarNames' 'classVarNames' 'classInstVarNames'
                                                'poolDictionaryNames' 'category' 'type' 'comment' 'stamp')
                                        DSClassCategoryChange #('oldCategory')
                                        DSClassCommentChange #('oldComment' 'oldStamp')
                                        DSClassCreatedChange #()
                                        DSClassNameChange #('oldName')
                                        DSClassRemovedChange #()
                                        DSClassSuperclassChange #('oldSuperclassName')
                                        DSClassTypeChange #('oldType')
                                        DSVarsChange #('oldVars')
                                                DSClassInstVarsChange #()
                                                DSClassVarsChange #()
                                                DSInstVarsChange #()
                                                DSSharedPoolVarsChange #()
                                DSMethodChange #('selector' 'meta' 'source' 'protocol' 'stamp')
                                        DSMethodAddedChange #()
                                        DSMethodProtocolChange #('oldProtocol')
                                        DSMethodRemovedChange #()
                                        DSMethodSelectorChange #('oldSelector')
                                        DSMethodSourceChange #('oldSource' 'oldStamp')
                        DSCompositeChange #()
                                DSChangeSequence #('changes')
                                        DSCompositeClassChange #('category')
                                                DSCompositeClassAddedChange #()
                                                DSCompositeClassDeletedChange #()
                        DSDoIt #('code')
```

Design
(some common sense and separation of concerns)

- A visitor does the actual apply
- The recording is separated
- Serialization is separated
- There are tests :)

Standalone object
(breaking rules)

- No references! Only plain data
- Easily serialized
- Easily manipulated
- Easily created and thrown away

Load and Apply
(an extra step!)

- Load into the image (from wherever)

  Now you can:

  - Edit it
  - Apply to the image atomically
  - Create an anti Delta

Revert
(neat!)

```
revert
	"Revert this delta by applying its anti Delta to the image."

	^self asAntiChange apply
```

```
asAntiChange
	"Return a new Delta that, when applied, reverts each change
	of this Delta in reverse order."

	| anti |
	anti := DSDelta name: 'undo ', self name.
	self propertiesDo: [:key :val | anti propertyAt: key put: val].
	self changes reverseDo: [:each |
		anti addChange: each asAntiChange].
	^anti
```

Revert of ”dirty” apply
(a clever trick!)

So what happens if you ”force” apply a Delta in another image? How can it revert?

Answer:

Record a sister Delta when applying!

Demo 2
(now watch him fail...)

- Harder stuff:
  - Create a Delta
  - Record more changes
  - Pick out a few, revert those
  - Revert all

Real world use
(can we use the darn thing?)

- Replace Changeset finally
- When MC fails (or is undesirable)
- Fork cross pollination
- Commit tool
- Advanced image changelog
- A base for other SCM solutions

Funky ideas
(cool stuff we probably should NOT do)

- More Change types:
  - Move method
  - Rename method
  - Refactorings?
- Delta reordering, ”commuting”

Streams
(Yeah!? What about that?)

- Chronological queue of Deltas
- Per package, person, image etc
- Categorized streams
- Publish/Discover/Subscribe
- ...can also be derived from MC repositories

Delta unit test debugging
(Advanced foot shootage)

- Put a halt in one of the revert tests
- Fix a bug when halted
- Do the usual proceed and be happy... or?

Conclusion: Meta programming is fun!

To do
(pretty please with sugar on top, help!)

- Cleanups (messy due to reorganisation)
- Tirade hooked in fully
- A DualDeltaSorter tool
- Getting it used!
- Streams...

Questions

(as if all is not 100% clear...)