cisa Centre for Intelligent Systems and their Applications

THE UNIVERSITY *of* EDINBURGH informatics

OK - Who Broke Everything ?

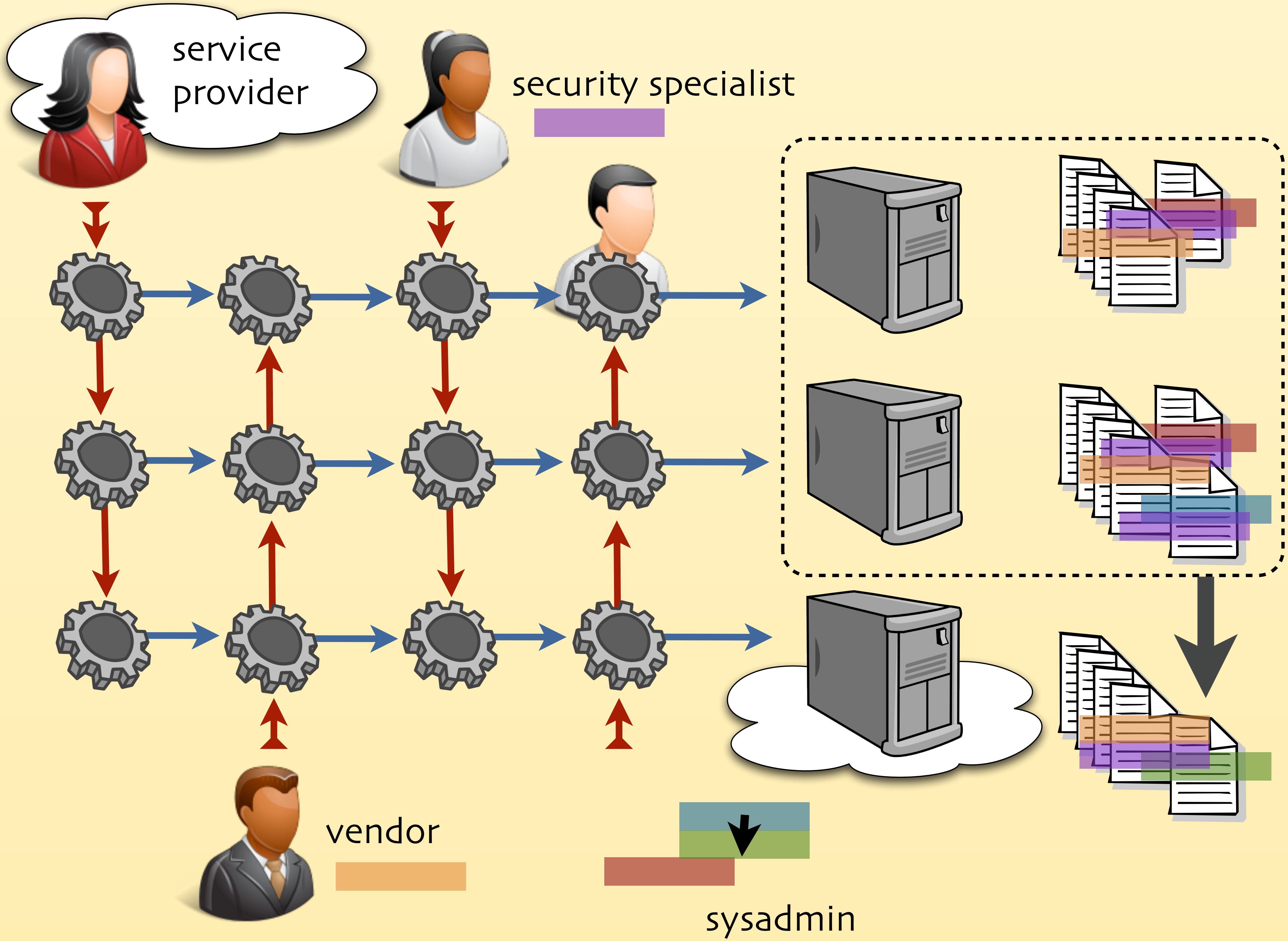
Provenance in system configuration languages

Paul Anderson

dcspaul@ed.ac.uk

http://homepages.inf.ed.ac.uk/dcspaul

http://homepages.inf.ed.ac.uk/dcspaul/publications/cisa-2013.pdf


service provider
security specialist
vendor
sysadmin

Aspect Composition

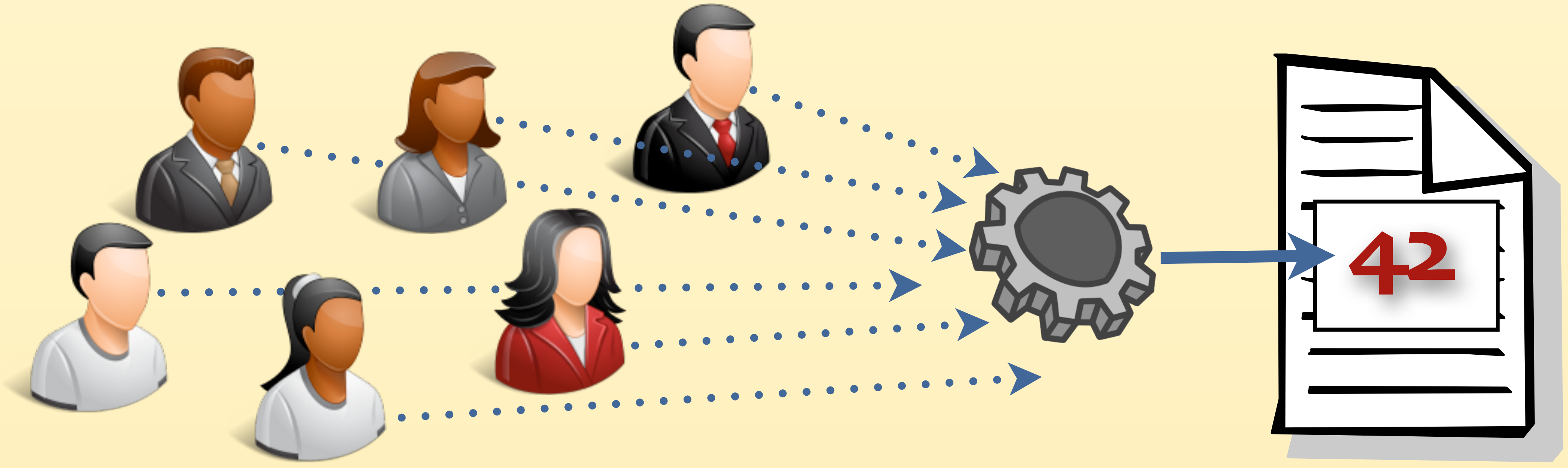
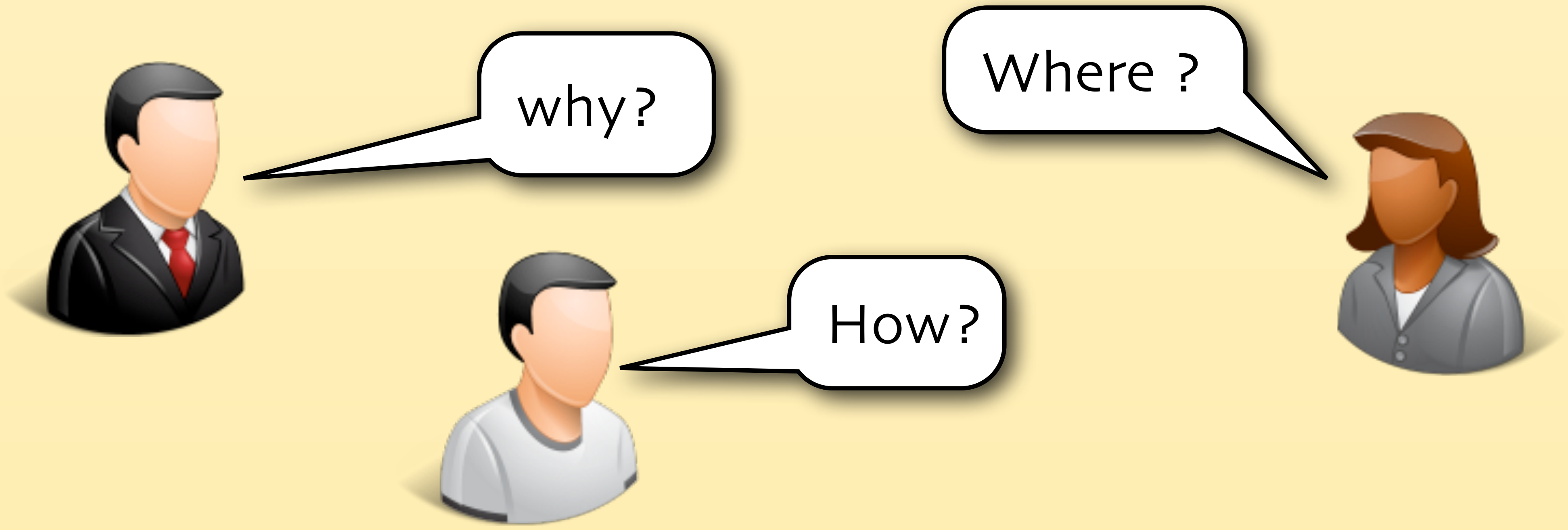
A configuration tool composes the independent "aspects" to form a consistent specification

- different tools support different languages and approaches

But the process can be complicated and involve a lot data

- simple order precedence
- more complex functions
- arbitrary constraints

So the "provenance" of the resulting configuration often unclear ....


42

why?
Where ?
How?

Provenance

Who is responsible for the fact that service X is running in the cloud when it shouldn't be? !

- many people may have specified rules contributing to this
- perhaps it was the fault of someone who said nothing at all!
  - i.e. there should have been a constraint preventing this

Were they all authorised to specify this?

Who needs to fix it?

- and how?

This has some analogies with provenance issues in databases

- James Cheney <jcheney@inf.ed.ac.uk> & I would like to explore this
- we have a Microsoft Phd award for this topic

A Typical Problem ...

Value Inheritance

```
class genericServer {
  timeServer = ts@reliable.com
  ... 742 more parameters ...
}
class widgetServer isa genericServer {
  ...
}
class salesServer isa widgetServer {
  ...
   ...
}
node serverA isa salesServer {
   ip = 1.2.3.4
   ...
}
```

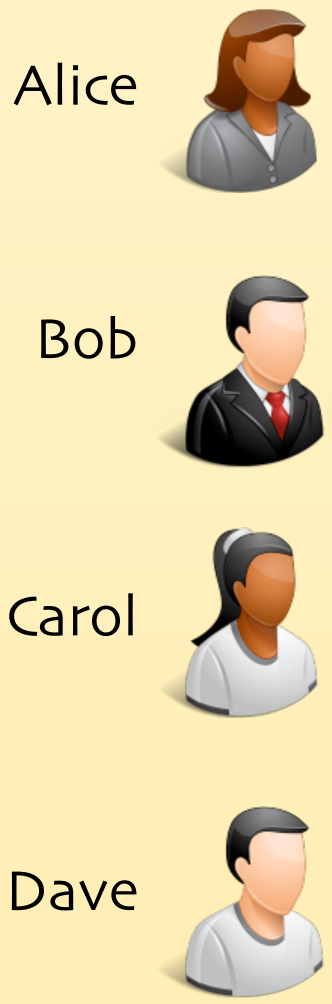
Alice Works For The Tool Vendor

```
class genericServer {
  timeServer = ts@reliable.com
  ... 742 more parameters ...
```

- Alice develops generic templates
- this one is for a generic server
- it specifies the default “timeserver”
- this is set to some reliable public service

```
}
node serverA isa salesServer {
   ip = 1.2.3.4
   ...
}
```

Bob Is The Senior Admin For widgets.com



```
class genericServer {
   timeServer = ts@reliable.com
   ... 742 more parameters ...
}
class widgetServer isa genericServer {
   ...
}
class salesServer isa widgetServer {

}
```

- Bob develops local templates
- these inherit from the generic ones
- Bob overrides some parameters
- but not the default timeserver

Carol Is The Admin For The Sales Dept

Alice

Bob

Carol

Dave

```
class genericServer {
```

- Carol inherits Bob's templates
- she overrides some parameters
- but not the default timeserver

```
class salesServer isa widgetServer {
   ...
    ...
}

node serverA isa salesServer {
    ip = 1.2.3.4
    ...
}
```

Dave Is The Technician

Alice

Bob

Carol

Dave

```
class genericServer {
  timeServer = ts@reliable.com
  ... 742 more parameters ...
}
```

- Dave configures the individual machines
- he assigns one of Carol’s templates
- overriding a few machine-specific values

```
    ...
}
node serverA isa salesServer {
    ip = 1.2.3.4
    ...
}
```

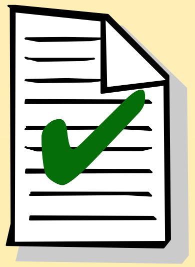
Carol Adds A Local Timeserver

```
class genericServer {
  timeServer = ts@reliable.com
  ... 742 more parameters ...
}

class widgetServer isa genericServer {
  ...
}

class salesServer isa widgetServer {
  timeServer = ts@sales.widget.com
  ...
}

node serverA isa salesServer {
   ip = 1.2.3.4
   ...
}
```



Alice Ships A New Template

Alice

Bob

Carol

Dave

```
class genericServer {
  timeServer = ts@unreliable.com
  ... 742 more parameters ...
}
class widgetServer isa genericServer {
  ...
}
class salesServer isa widgetServer {
  timeServer = ts@sales.widget.com
  ...
}
node serverA isa salesServer {
   ip = 1.2.3.4
   ...
}
```

Carol Withdraws Her Change

Alice

Bob

Carol

Dave

```
class genericServer {
  timeServer = ts@unreliable.com
  ... 742 more parameters ...
}
class widgetServer isa genericServer {
  ...
}
class salesServer isa widgetServer {
~~  timeServer = ts@sales.widget.com~~
  ...
}
node serverA isa salesServer {
   ip = 1.2.3.4
   ...
}
```

Whose “Fault” Is This?

Dave’s server broke and he got the blame from the users

- in fact, all of the machines in the Sales Department are broken!
- but he says he didn’t change anything at all

Carol says she just put the parameter back to the default

- so it can’t be her fault - this is exactly the same as it was before

Bob says he carefully checked the new default configuration

- in fact, he ran some regression tests and the new configuration produced exactly the same results as the old one on all of the Sales Department machines

Alice says that she changed this default ages ago

- and it is up to the users to check these changes are appropriate
- although it is Alice’s value which appears in the final configuration

Who Should Fix It? And How?

Alice probably isn't going to change this

- she presumably had a good reason for the new value
- and she doesn't work for us anyway, so she may break it again ...

Dave doesn't want to set it on his individual machines

- although he might do this as an interim fix!
- which will of course cause problems later, if it doesn't get removed

Carol just wants the same value as the rest of the company

- although she could make an interim fix too

But it is probably Bob who needs to make a company-wide change ?

- even though he was not responsible for any of the changes which exposed the problem

Tracking Provenance

We need to know who authored what

- relating source text diffs to semantic changes is not reliable

Every value must have a corresponding provenance expression

- the language needs a "provenance semantics"
  as well as the conventional "value semantics"
- there may be multiple different interpretations for different purposes

The provenance tends to be "explosive"

- "everyone had their fingers in this"
- we may need to evaluate (for example) both branches of a conditional

This needs to be implemented in the configuration compiler

A Provenance Semantics ?

`{Alice}`	`X=2`
`{Bob}`	`Y=3`
`{Carol}`	`if X==2 then`
`{Dave}`	`  Y=4`
`{Carol}`	`else`
`{Erin}`	`  Y=5`
`{Carol}`	`fi`

The value of Y is 4

Because Dave said so

But Alice had a say in this

If she changed her line, the result would be different

So did Carol

P = {D,A,C} ?

But what about Erin?

If her value was 4, then it would no longer matter what Alice said!

Some Questions

Perhaps we need multiple notions of provenance for different purposes?

- using the result for security (allow/disallow changes) ?

Is the history is important to understanding ?

- when Alice changed the default value, the configuration started to "smell bad", even though there was no immediate consequences
- even though the specification is entirely declarative, it may be useful to know "how we got here"

Perhaps we can assign some degree of "robustness" ?

- the above configuration is less robust in some sense, because it is more likely to break when things change
- is it right that things should break if I back out a change ?
- can I be warned when that situation is likely to occur ?

cisa Centre for Intelligent Systems and their Applications

THE UNIVERSITY *of* EDINBURGH informatics

OK - Who Broke Everything ?

Provenance in system configuration languages

Paul Anderson

dcspaul@ed.ac.uk

http://homepages.inf.ed.ac.uk/dcspaul

http://homepages.inf.ed.ac.uk/dcspaul/publications/cisa-2013.pdf