Refactoring for End Users

Kathryn T. Stolee and Sebastian Elbaum
University of Nebraska–Lincoln

May 22, 2011

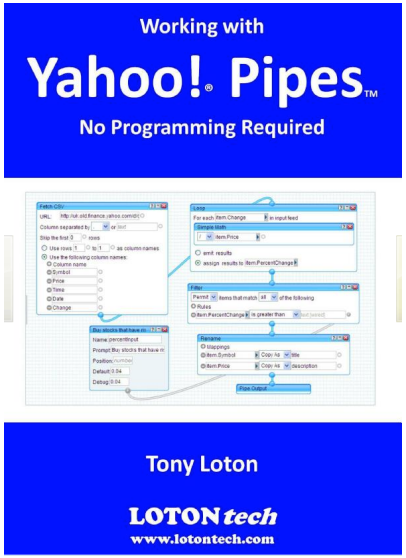
In this talk

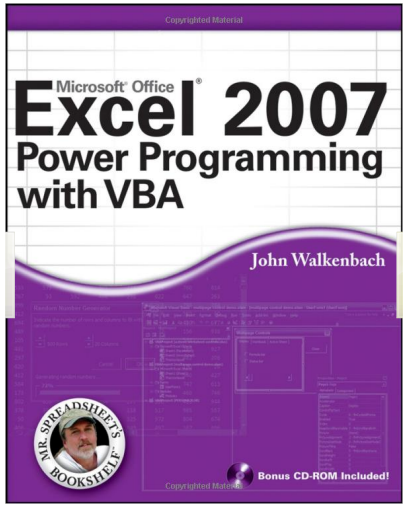
- End-user software engineering
- Refactoring for web mashups
- Mashup languages create opportunities to extend refactorings
- Speculate how refactoring adapts to other end user domains

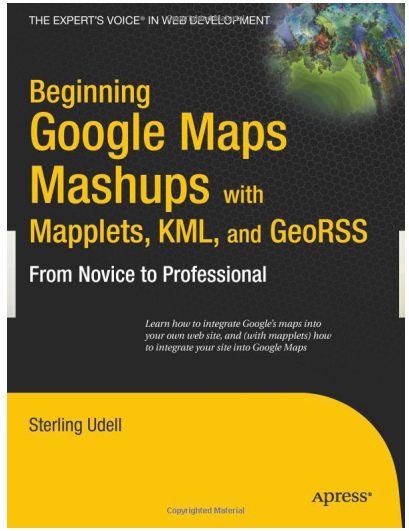
In this talk

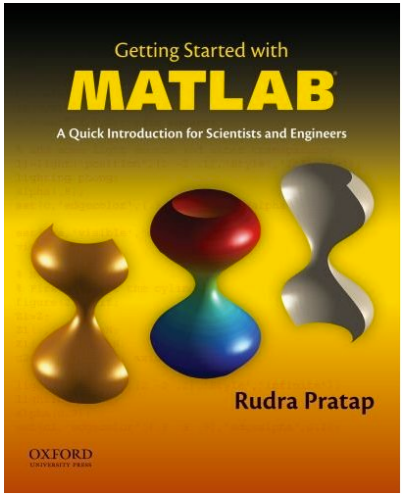
- End-user software engineering
- Refactoring for web mashups
- Mashup languages create opportunities to extend refactorings
- Speculate how refactoring adapts to other end user domains

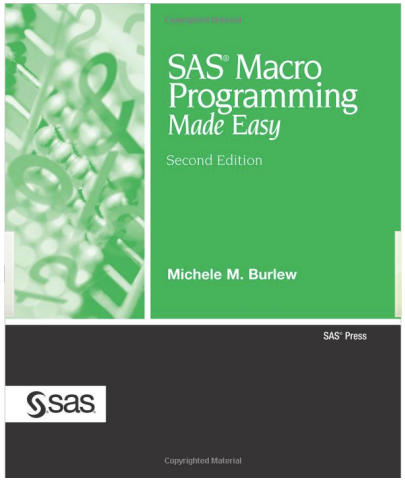
End-User Programming

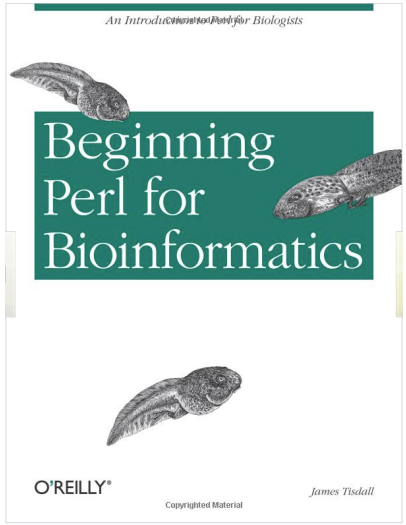


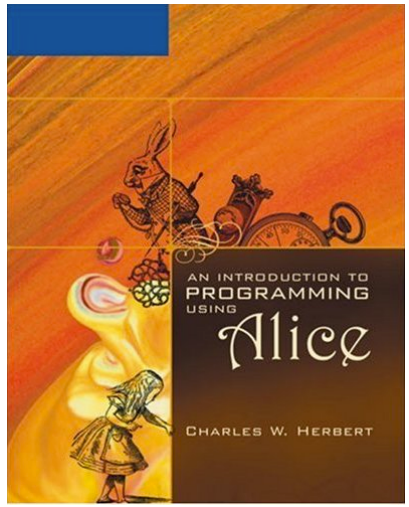












About End-User Programmers

- **There are many** (an estimated 55 million in 2005)
- **Their jobs are diverse** (e.g., accountants, biologists, teachers)
- **They create useful software**, for example:
  - Macros save hundreds of hours in data entry
  - Spreadsheets allow for efficient data storage and computation

About End-User Programmers

- **There are many** (an estimated 55 million in 2005)
- **Their jobs are diverse** (e.g., accountants, biologists, teachers)
- **They create useful software**, for example:
  - Macros save hundreds of hours in data entry
  - Spreadsheets allow for efficient data storage and computation

Undermines common assumptions*

- Software is mostly created by professionals
- End users just need good user interface

*Credit to Mary Shaw

Qualitative Differences Between Professional and End-User Programmers

Software Engineering Activity	**Professional SE**	**End-user SE**
Specifications & Requirements	*explicit*	*implicit*
Reuse	*planned*	*unplanned*
Testing and Verification	*cautious*	*overconfident*
Debugging	*systematic*	*opportunistic*

* Credit to Ko, A. et al., The State-of-the-Art in End-User Software Engineering

Challenges in End-User Programming

- **Faults**
  An error in a spreadsheet formula cost a Texas oil firm millions of dollars in an acquisition deal

  Panko, R. “What We Know About Spreadsheet Errors.” Journal of End User Computing Spring 1998.
- **Design**
  Decomposing a problem is difficult for end-user programmers.

  Chambers, C. and Scaffidi, C. “Struggling to Excel: A Field Study of Challenges Faced by Spreadsheet Users.” VL/HCC 2010.
- **Understanding**
  Most learning barriers regarding program understanding were unsurmountable for new programmers.

  Ko, A.J., Myers, B.A., Aung, H.H. “Six Learning Barriers in End-User Programming Systems.” VL/HCC. 2004.
- **Maintenance**, **Debugging**, **Testing**, . . .

Challenges in End-User Programming

- **Faults**
  An error in a spreadsheet formula cost a Texas oil firm millions of dollars in an acquisition deal

  Panko, R. “What We Know About Spreadsheet Errors.” Journal of End User Computing Spring 1998.
- **Design**
  Decomposing a problem is difficult for end-user programmers.

  Chambers, C. and Scaffidi, C. “Struggling to Excel: A Field Study of Challenges Faced by Spreadsheet Users.” VL/HCC 2010.
- **Understanding**
  Most learning barriers regarding program understanding were unsurmountable for new programmers.

  Ko, A.J., Myers, B.A., Aung, H.H. “Six Learning Barriers in End-User Programming Systems.” VL/HCC. 2004.
- **Maintenance**, **Debugging**, **Testing**, . . .

Generally, these tasks are unsupported in EUP environments

End User Software Engineering

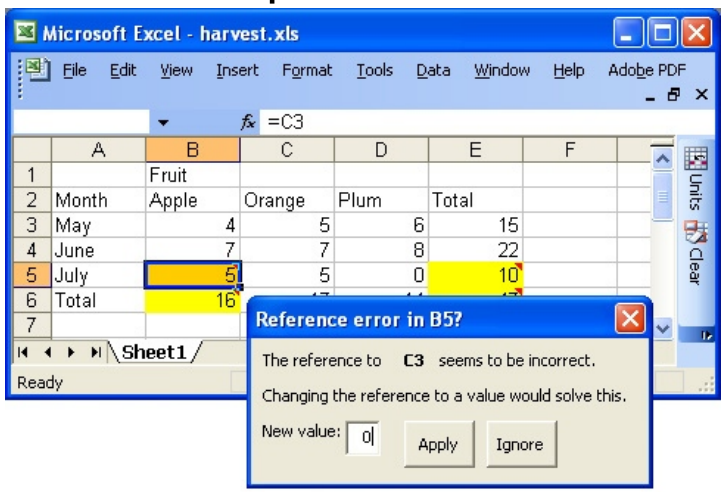
Goal: Better development support for end-user programmers*

- Without requiring training or even *interest* in software engineering
- A holistic approach: no separated tasks, modes, or tools

* Credit to Margaret Burnett

Current End-User Research

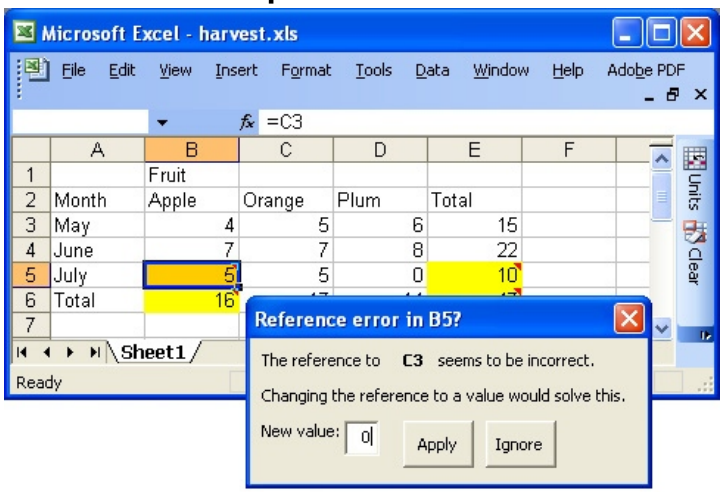
UCheck Spreadsheet Errors



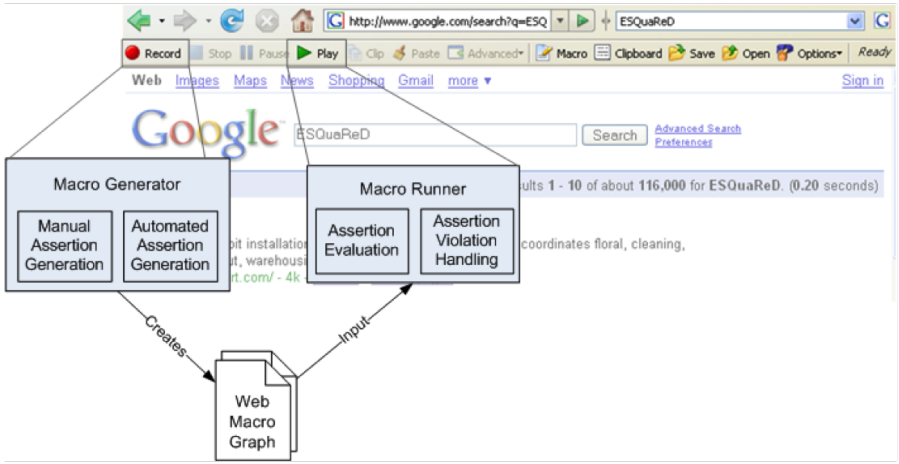
Credit: EUSES Consortium `http://eusesconsortium.org/`

Current End-User Research

UCheck Spreadsheet Errors



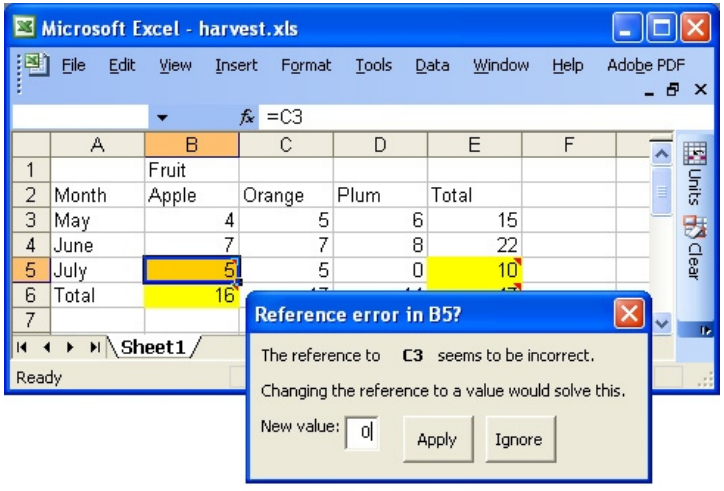
Robofox Assertion Generation



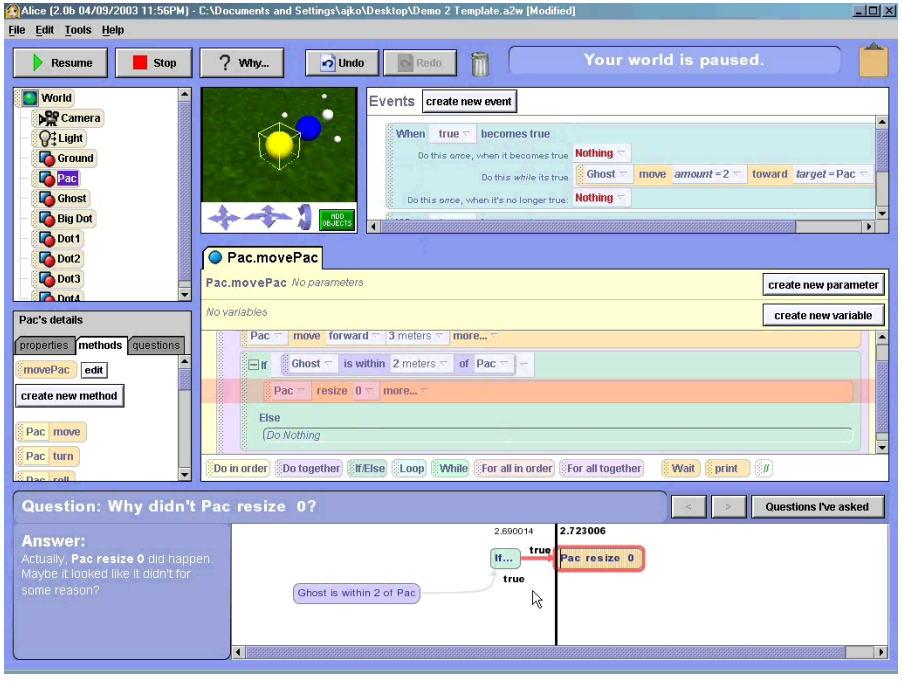
Credit: EUSES Consortium `http://eusesconsortium.org/`

Current End-User Research

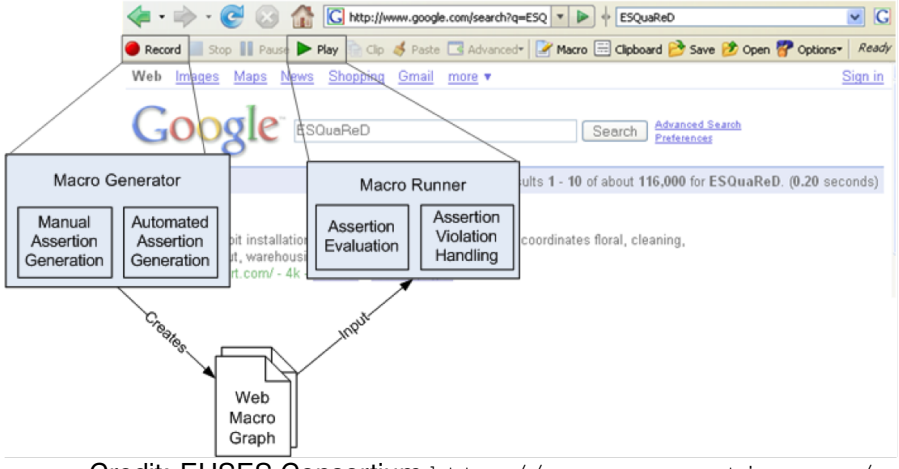
UCheck Spreadsheet Errors



Whyline Debugging



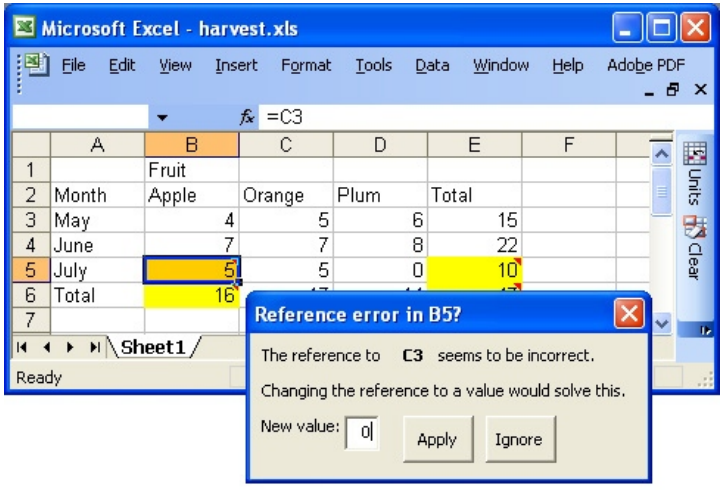
Robofox Assertion Generation



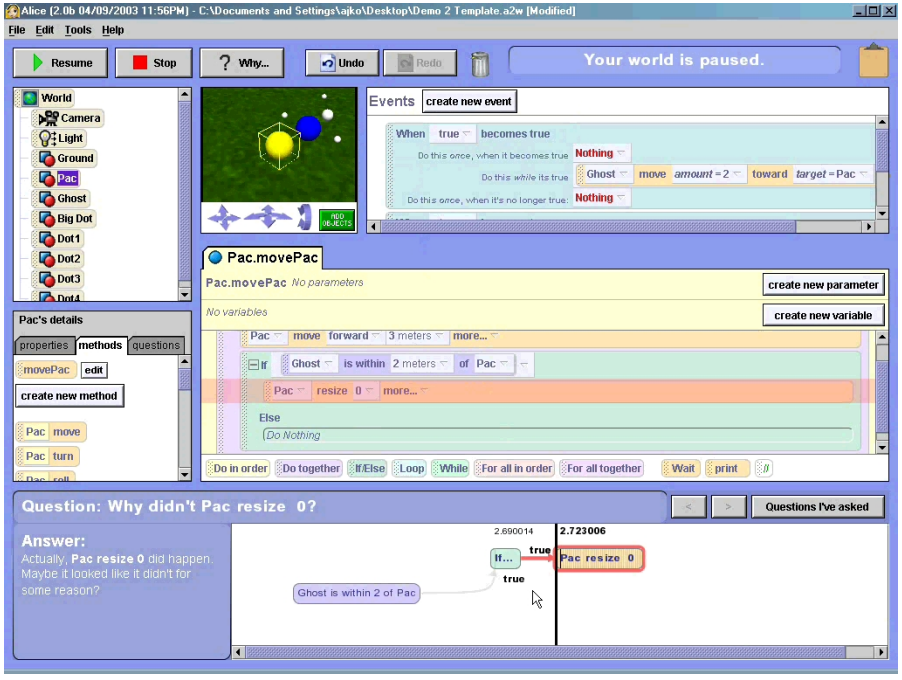
Credit: EUSES Consortium `http://eusesconsortium.org/`

Current End-User Research

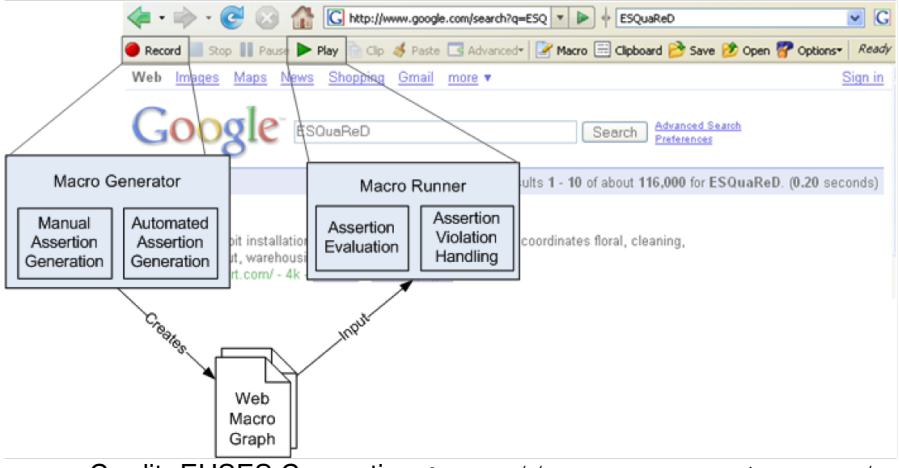
UCheck Spreadsheet Errors



Whyline Debugging

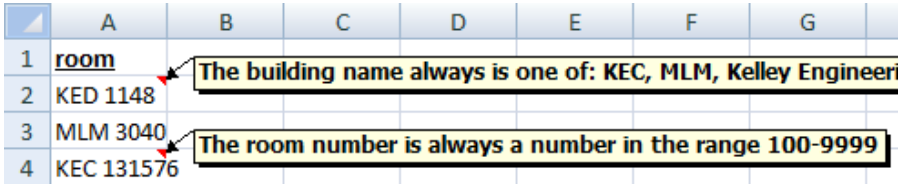


Robofox Assertion Generation



Credit: EUSES Consortium `http://eusesconsortium.org/`

Topes Data Validation



In this talk

- End-user software engineering
- Refactoring for web mashups
- Mashup languages create opportunities to extend refactorings
- Speculate how refactoring adapts to other end user domains

Why Web Mashups?

Web Mashups

Applications that compose and manipulate existing data sources or services to create new data or service.

Why Study Mashups?

- Many environments (e.g., Apatar, DERI Pipes, IBM Mashup Center, Kivati, Yahoo! Pipes, . . .)
- Potential impact
- Communities

Why Refactoring for Mashups?

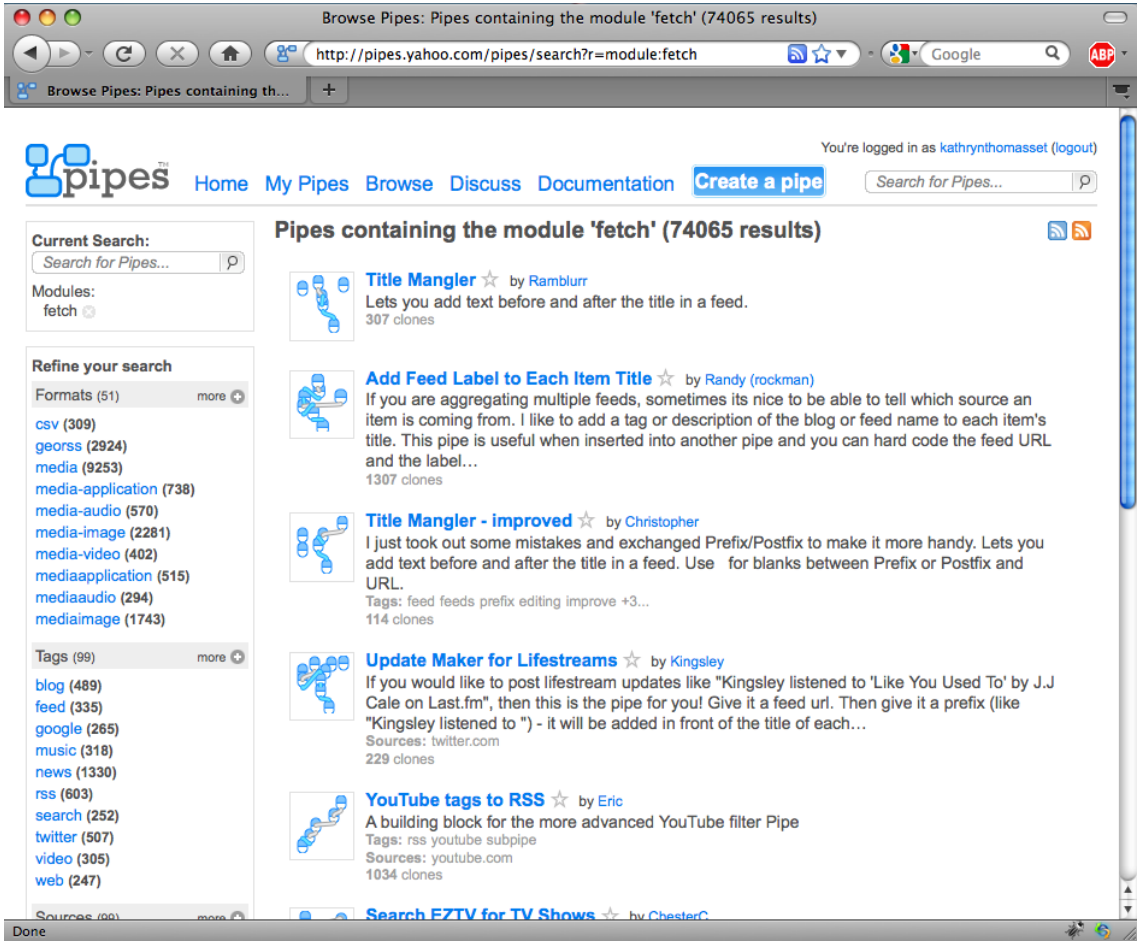
- Mashup programs are littered with code smells
- Smells matter to end users

  Stolee, K.T. and Elbaum, S. Refactoring Pipe-like Mashups for End-User Programmers. ICSE 2011
- Mashup languages are hard for end users to understand

  Zang, N. and Rosson, M.B. Playing with information: How end users think about and integrate dynamic data. VL/HCC 2009
- Maintenance in mashups is common: nearly 24% of Yahoo! Pipes mashups were modified after they were made public
- Reuse then adaptation is common: over 54% of pipes have been cloned, yet only 5% have an exact match

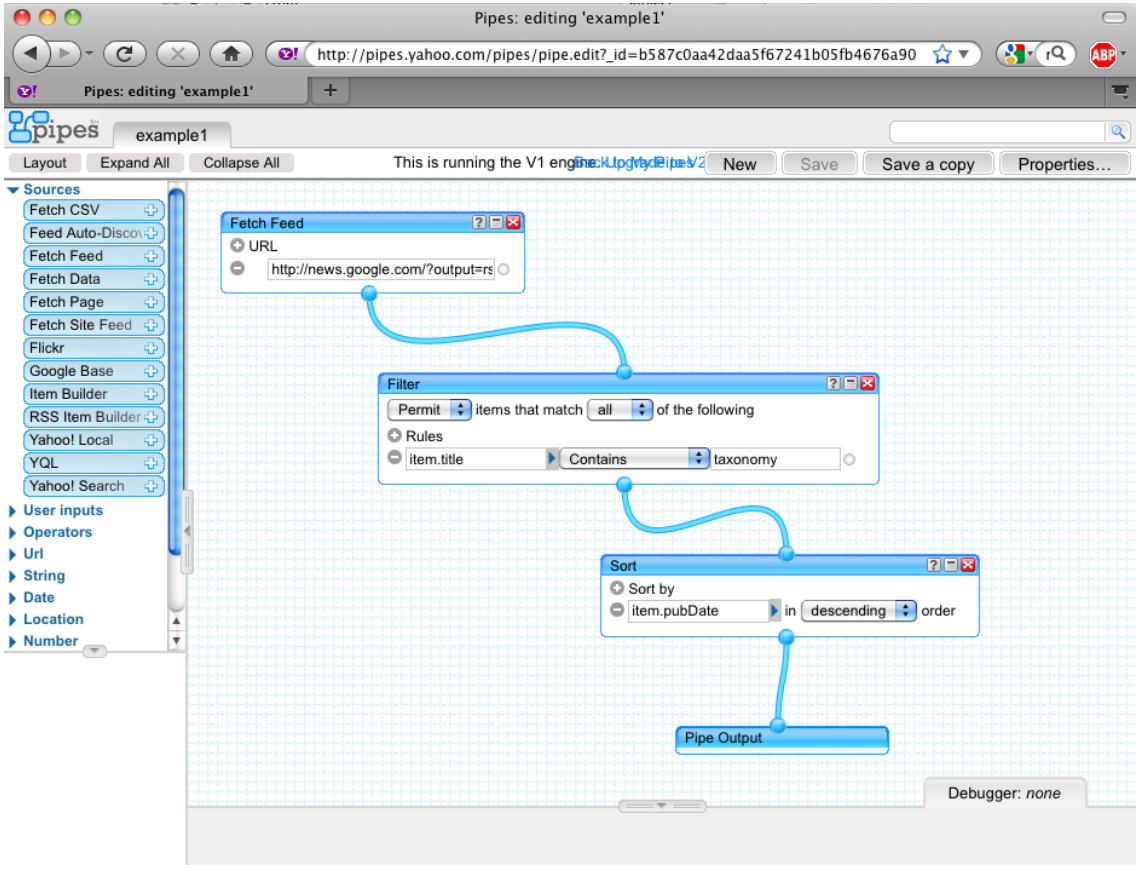
  Stolee, K.T., Elbaum, S., and Sarma, A. End-User Programmers and their Communities: An Artifact-based Analysis. ESEM 2011

About Yahoo! Pipes



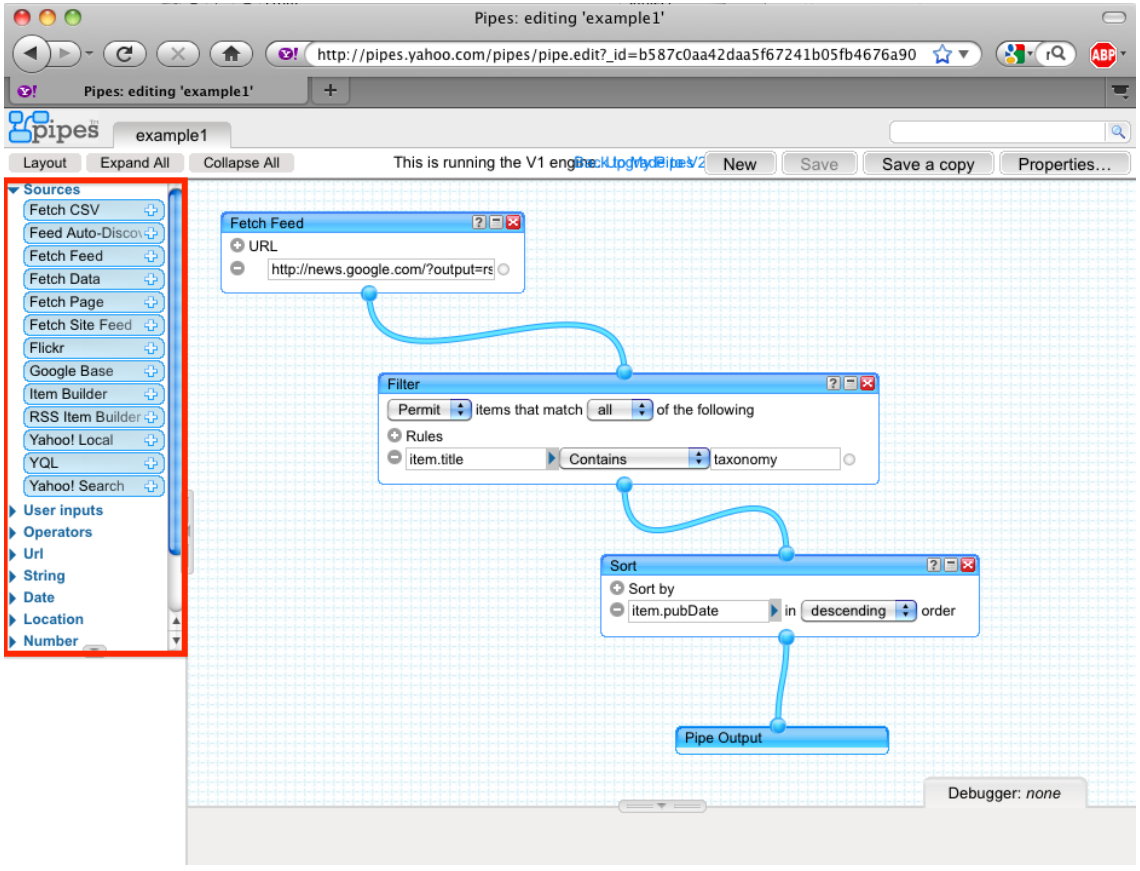
- Over 90,000 authors have created mashups
- 5,000,000+ pipes executed every day
- **Large public repository of sample code**

About Yahoo! Pipes



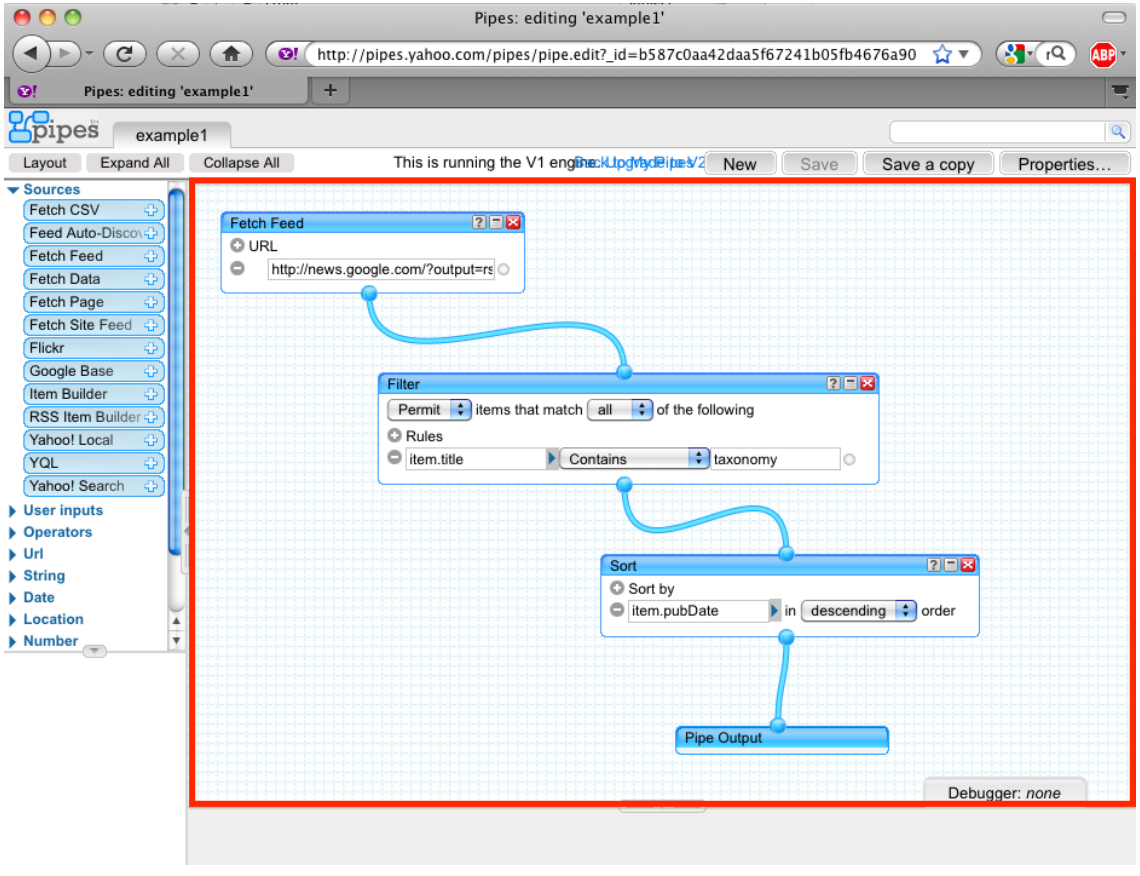
- Visual mashup creation environment
- Within a browser
- Drag and drop interface
- Constrained, yet flexible language

About Yahoo! Pipes



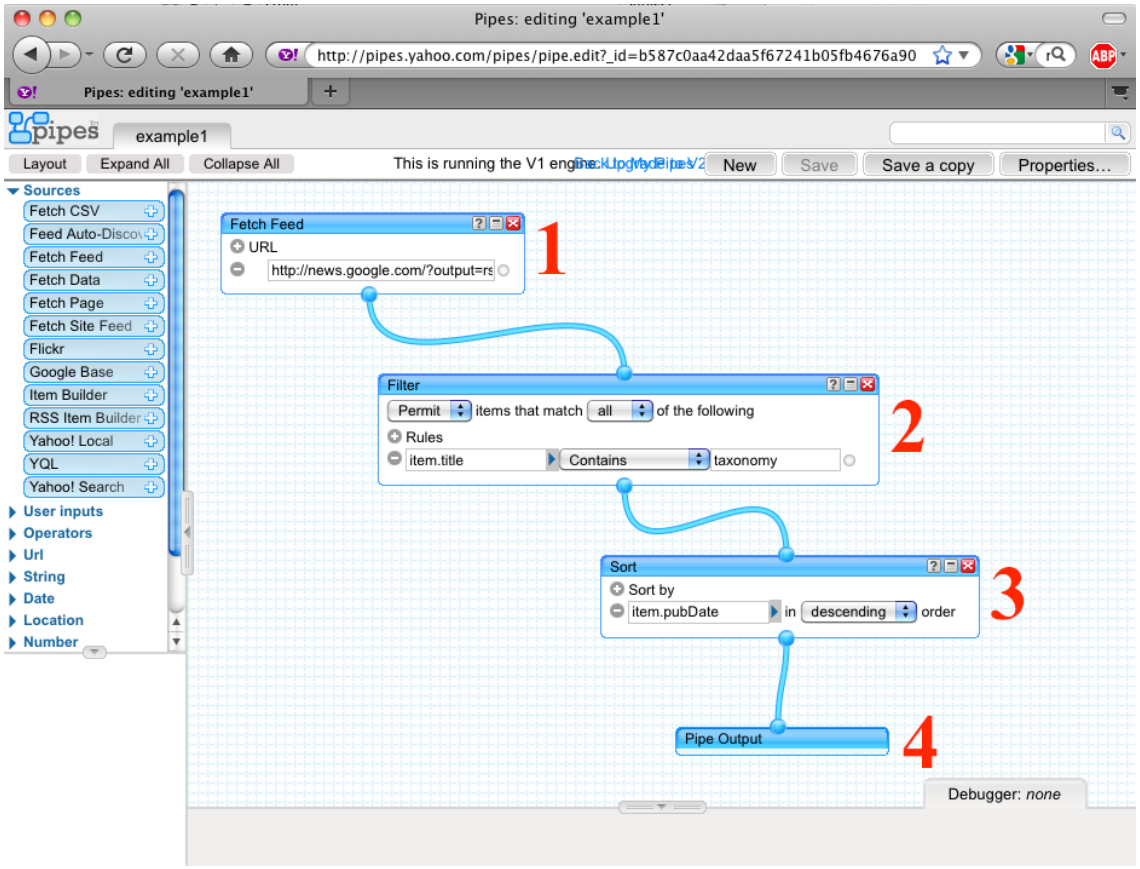
- Visual mashup creation environment
- Within a browser
- Drag and drop interface
- Constrained, yet flexible language

About Yahoo! Pipes



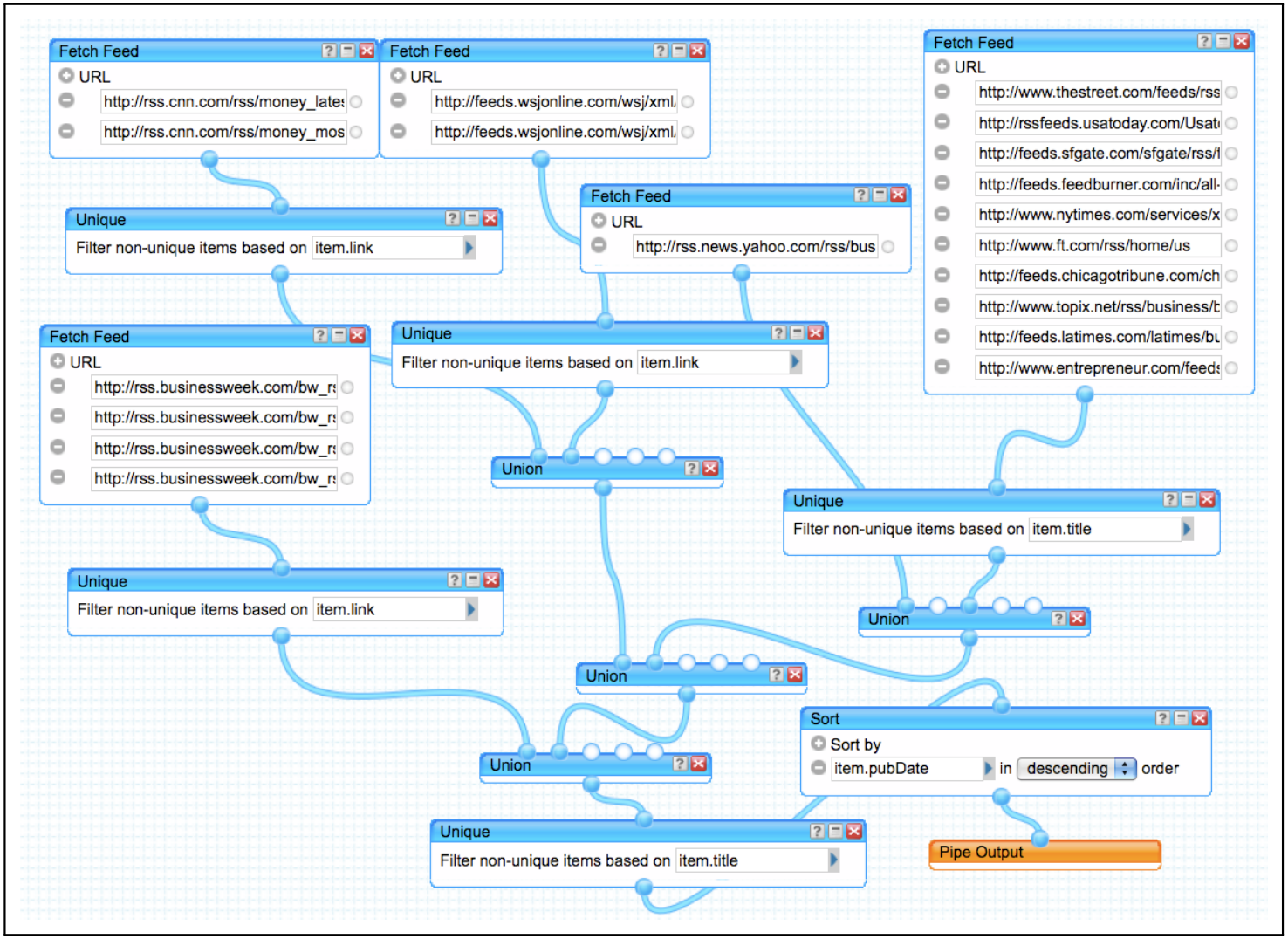
- Visual mashup creation environment
- Within a browser
- Drag and drop interface
- Constrained, yet flexible language

About Yahoo! Pipes

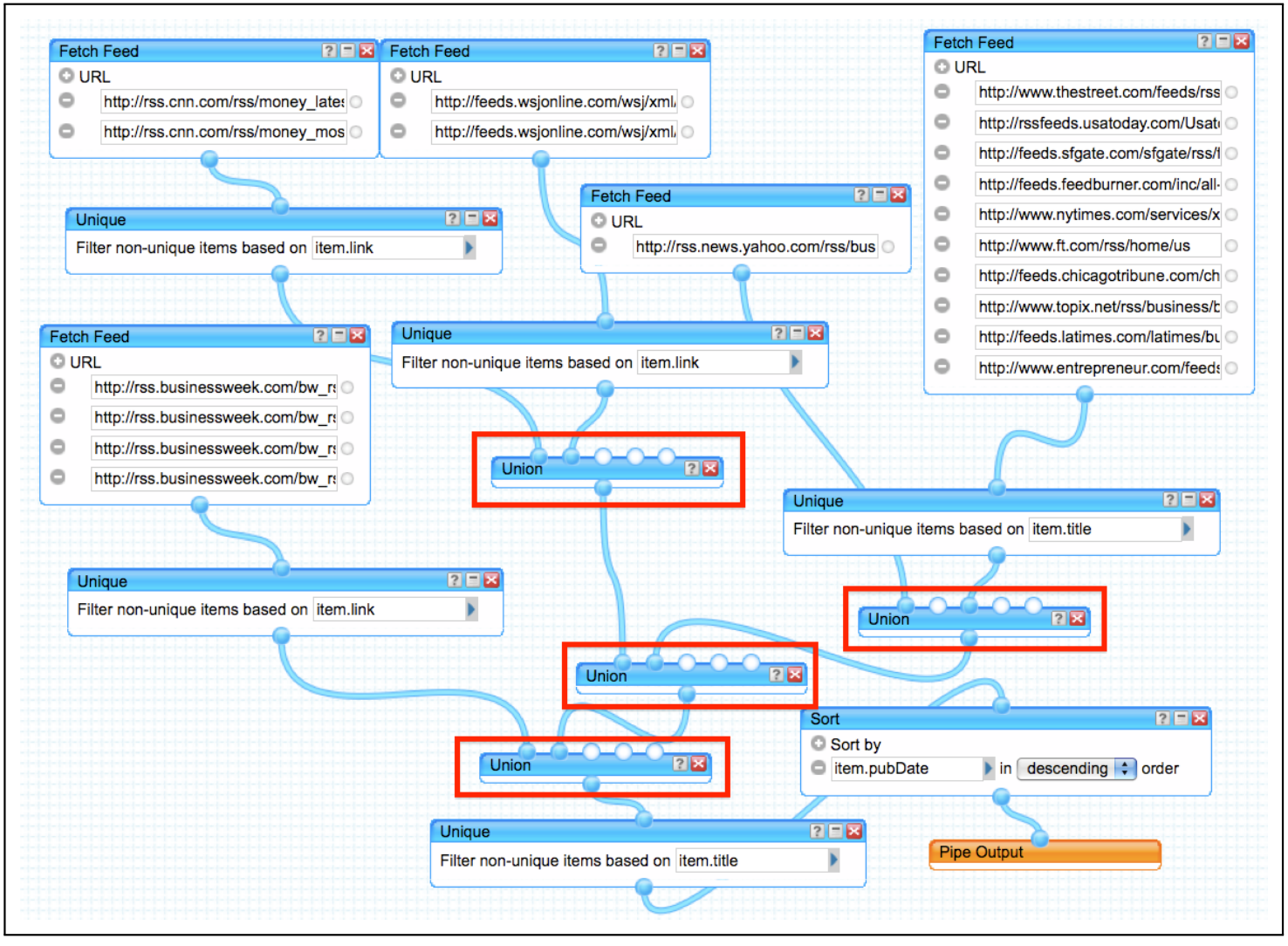


- Visual mashup creation environment
- Within a browser
- Drag and drop interface
- Constrained, yet flexible language

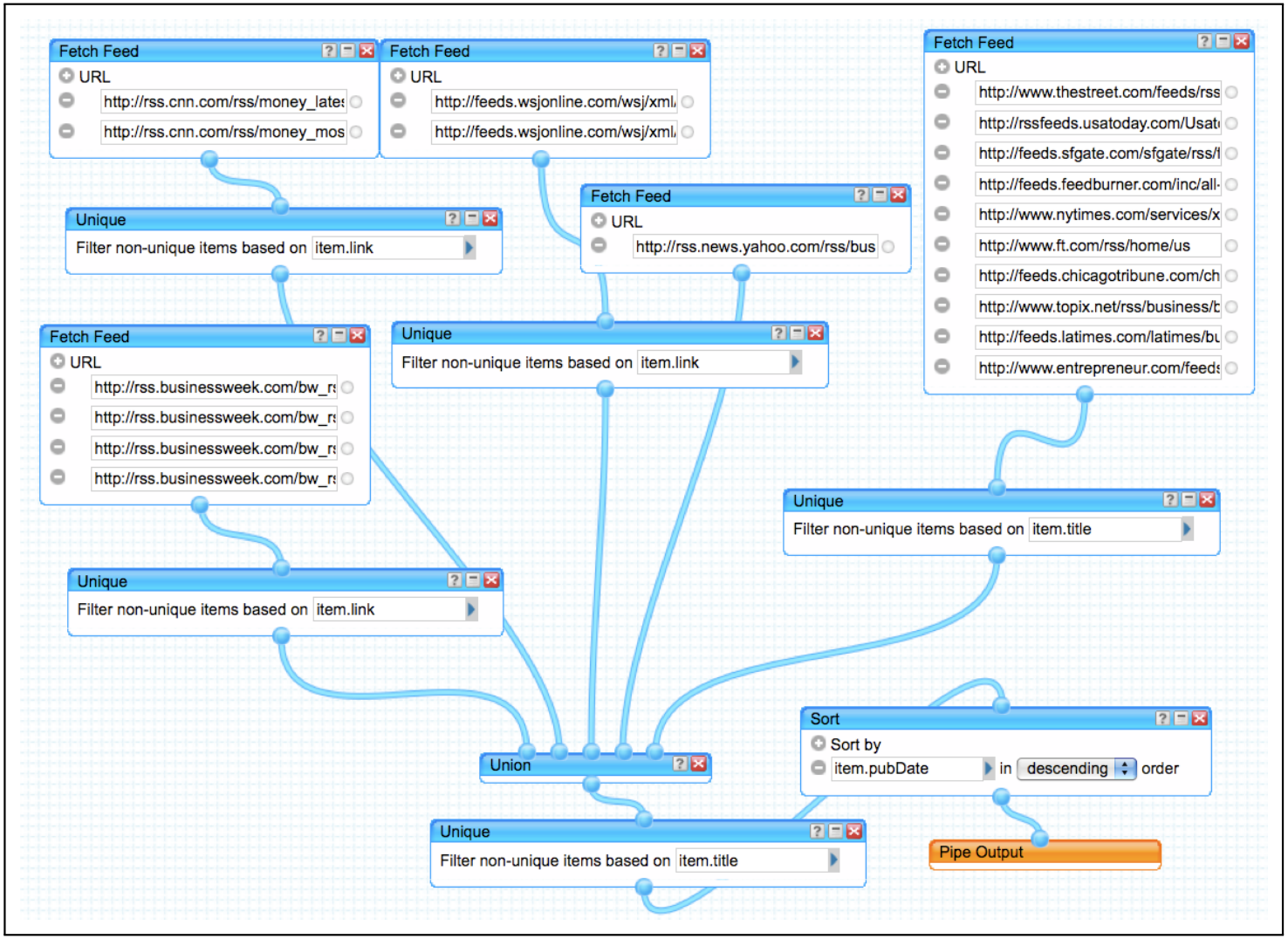
Motivating Example



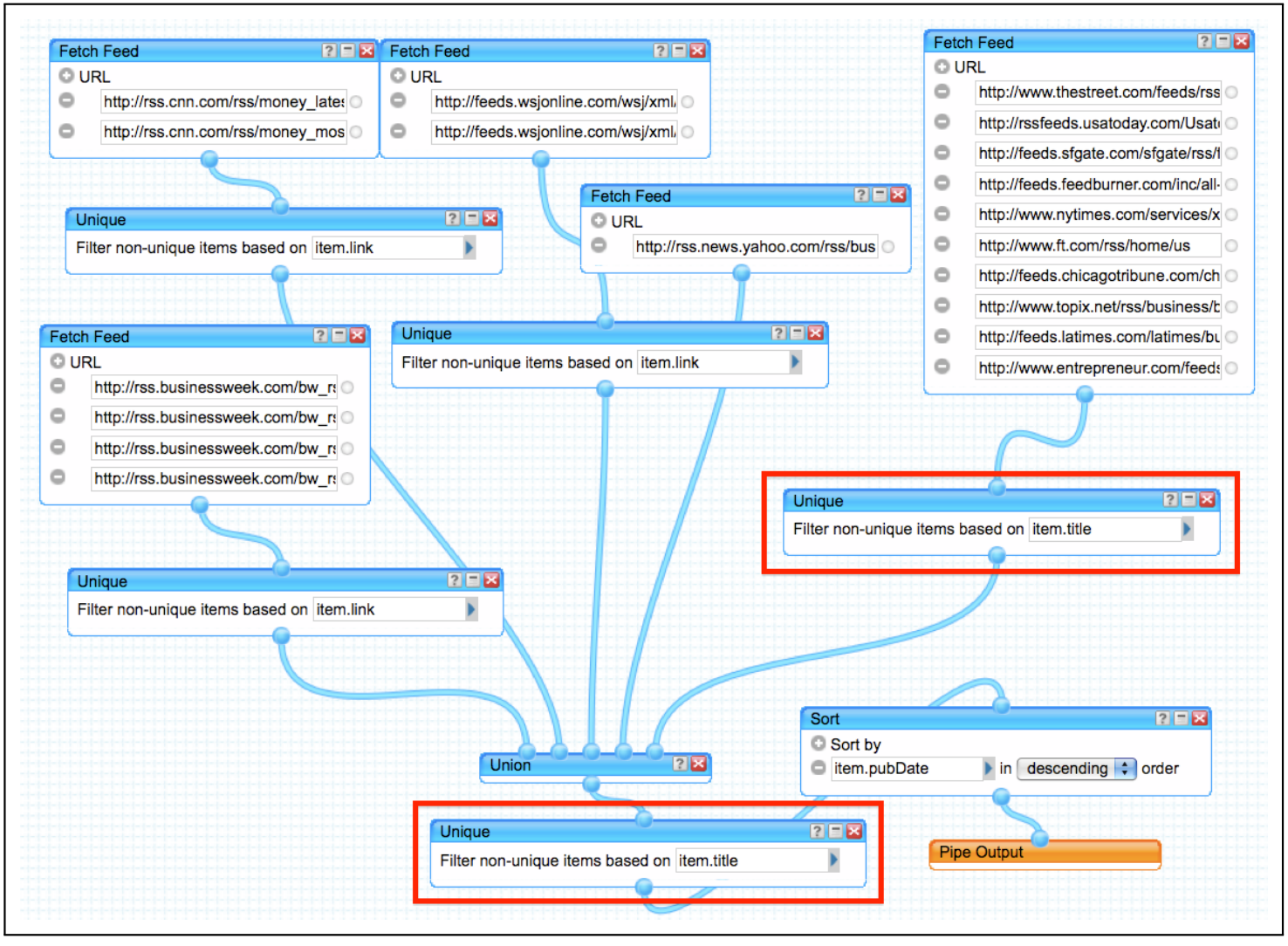
Motivating Example



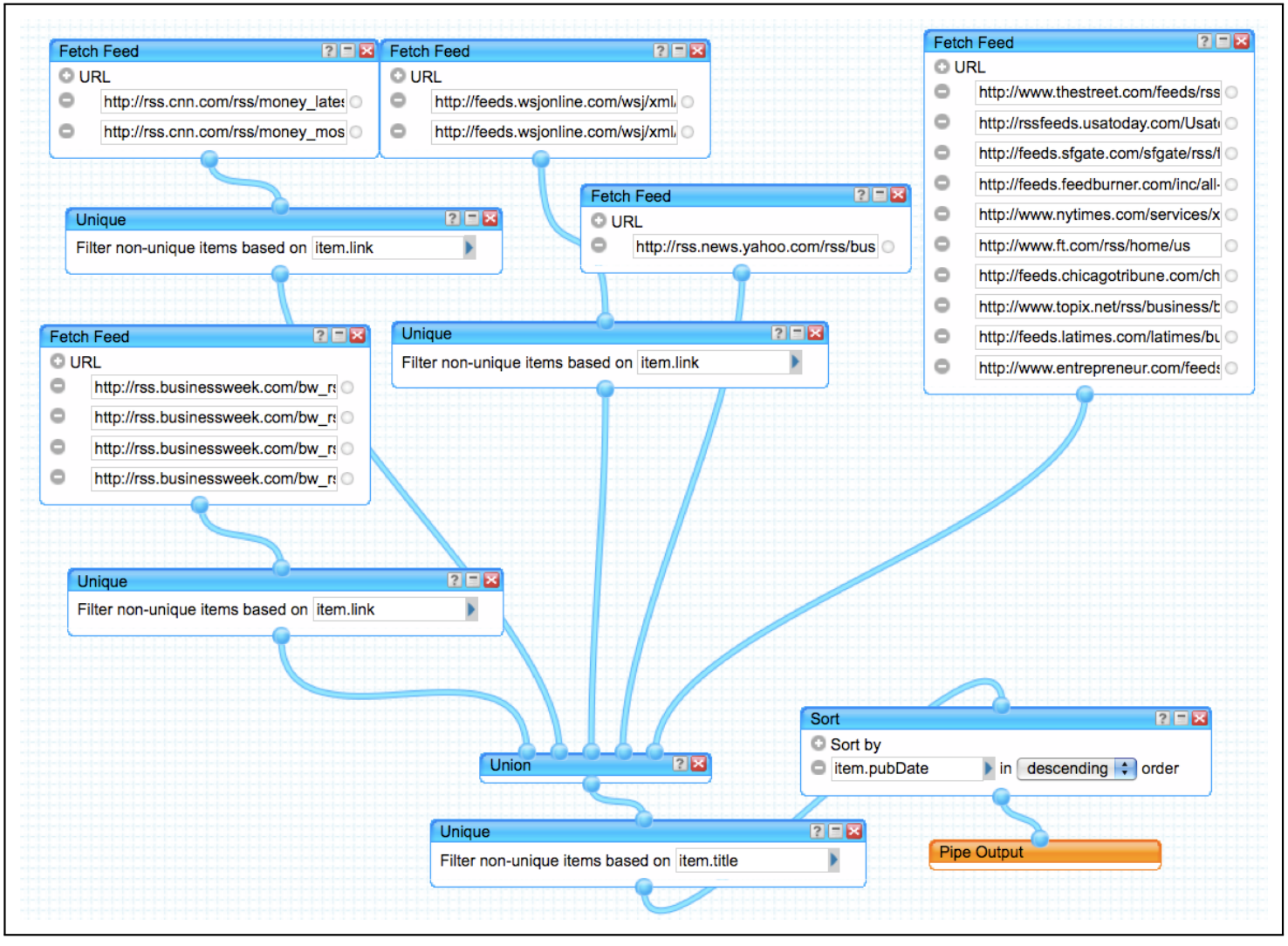
Motivating Example



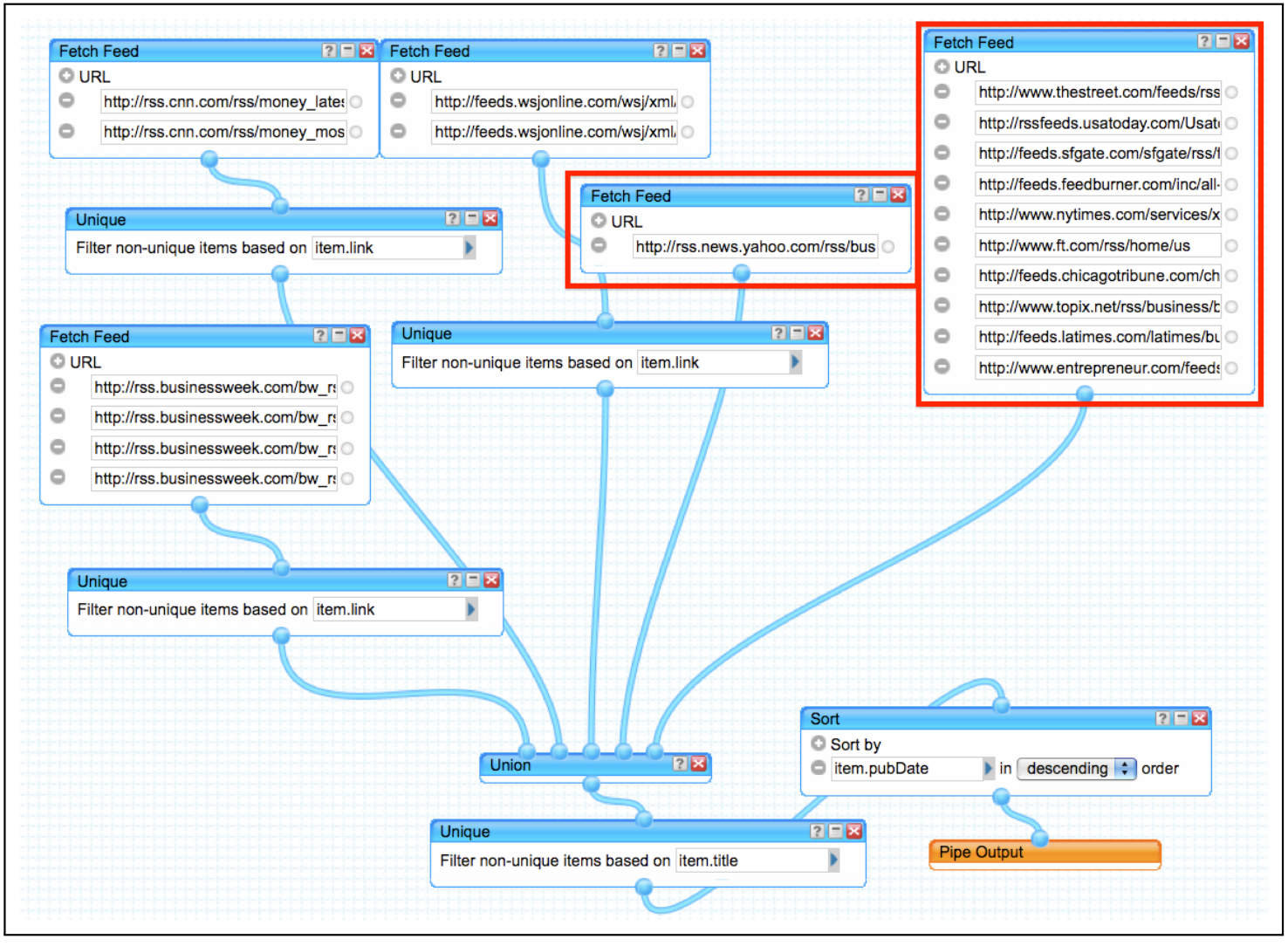
Motivating Example



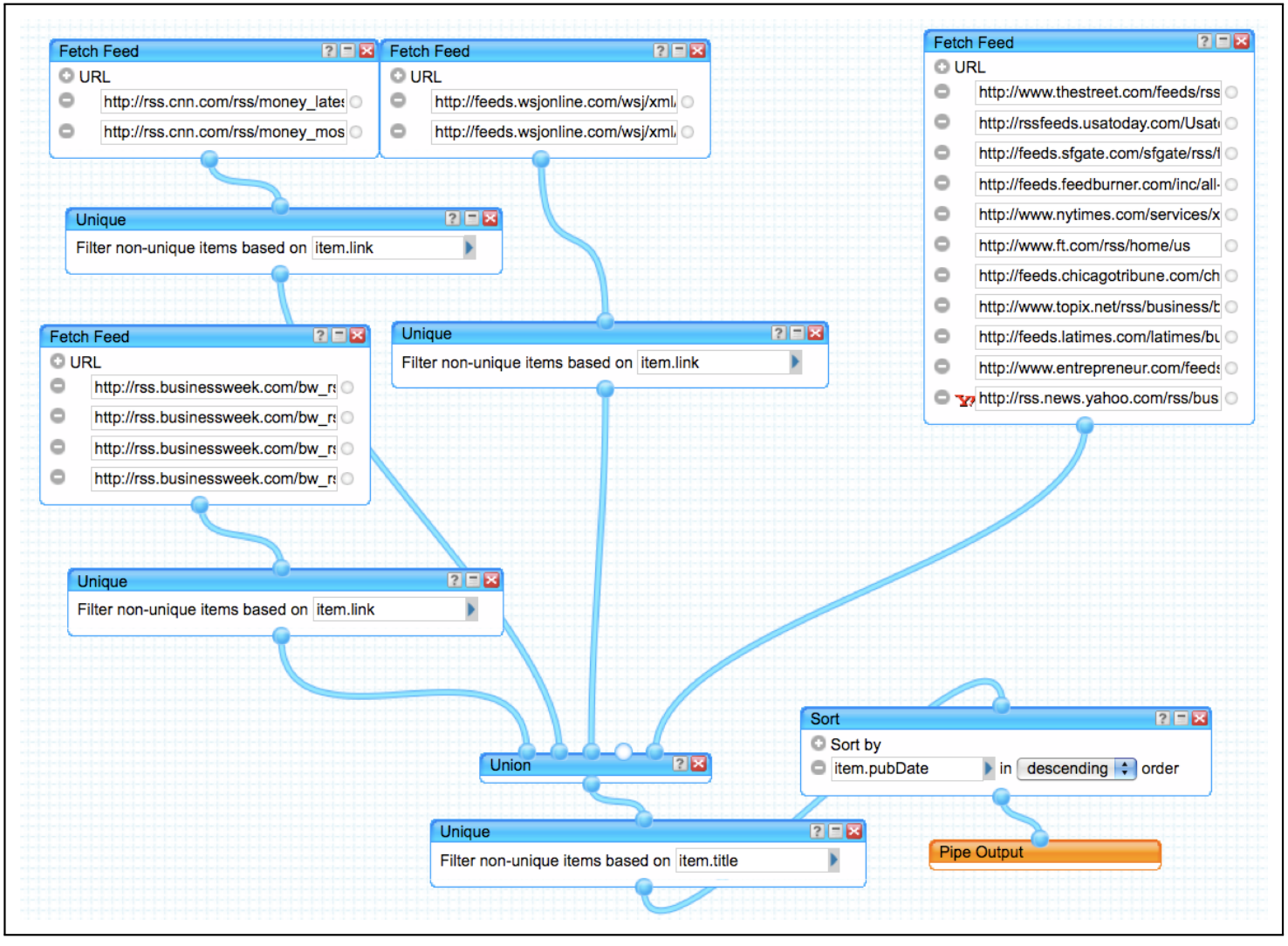
Motivating Example



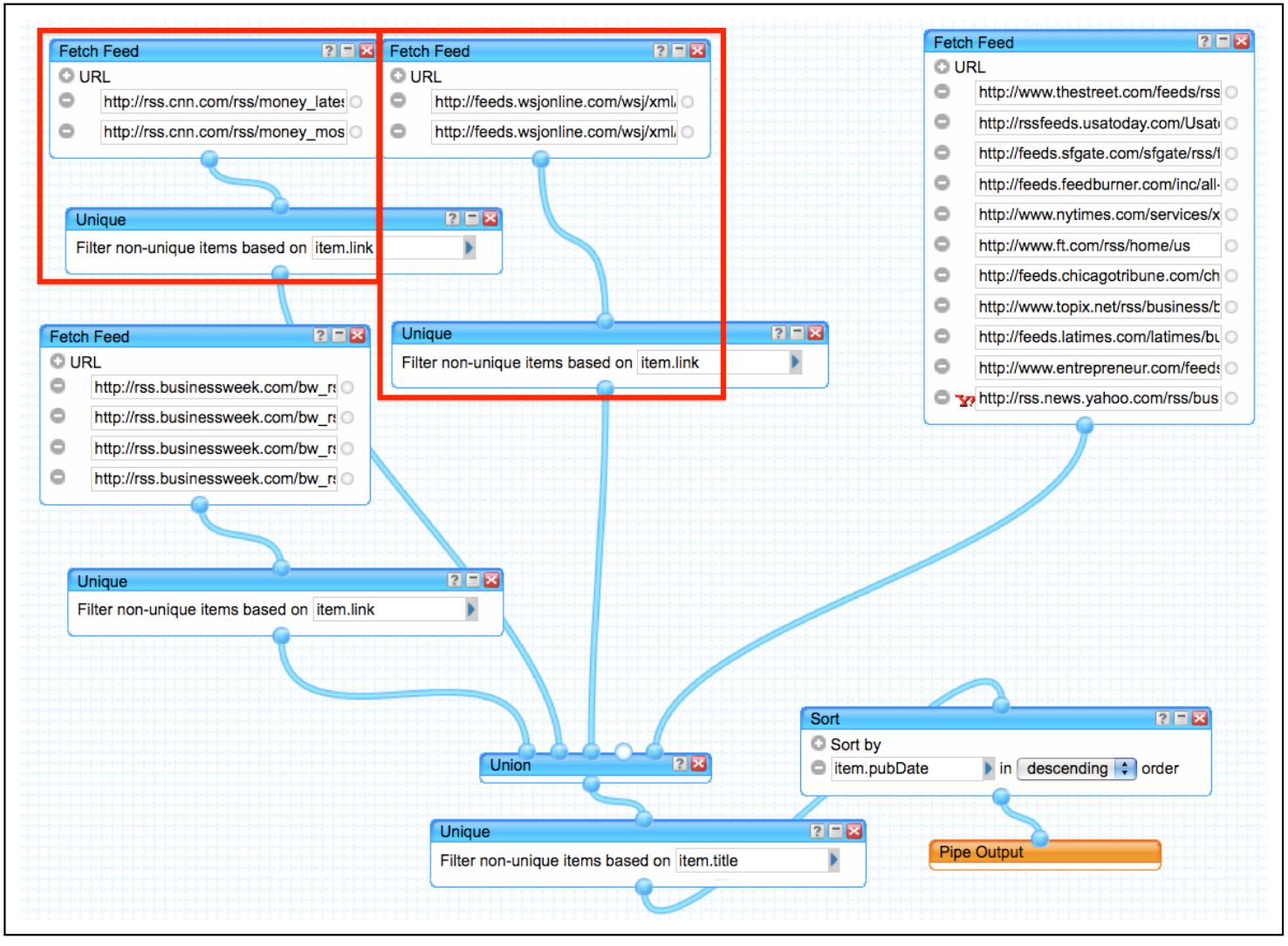
Motivating Example



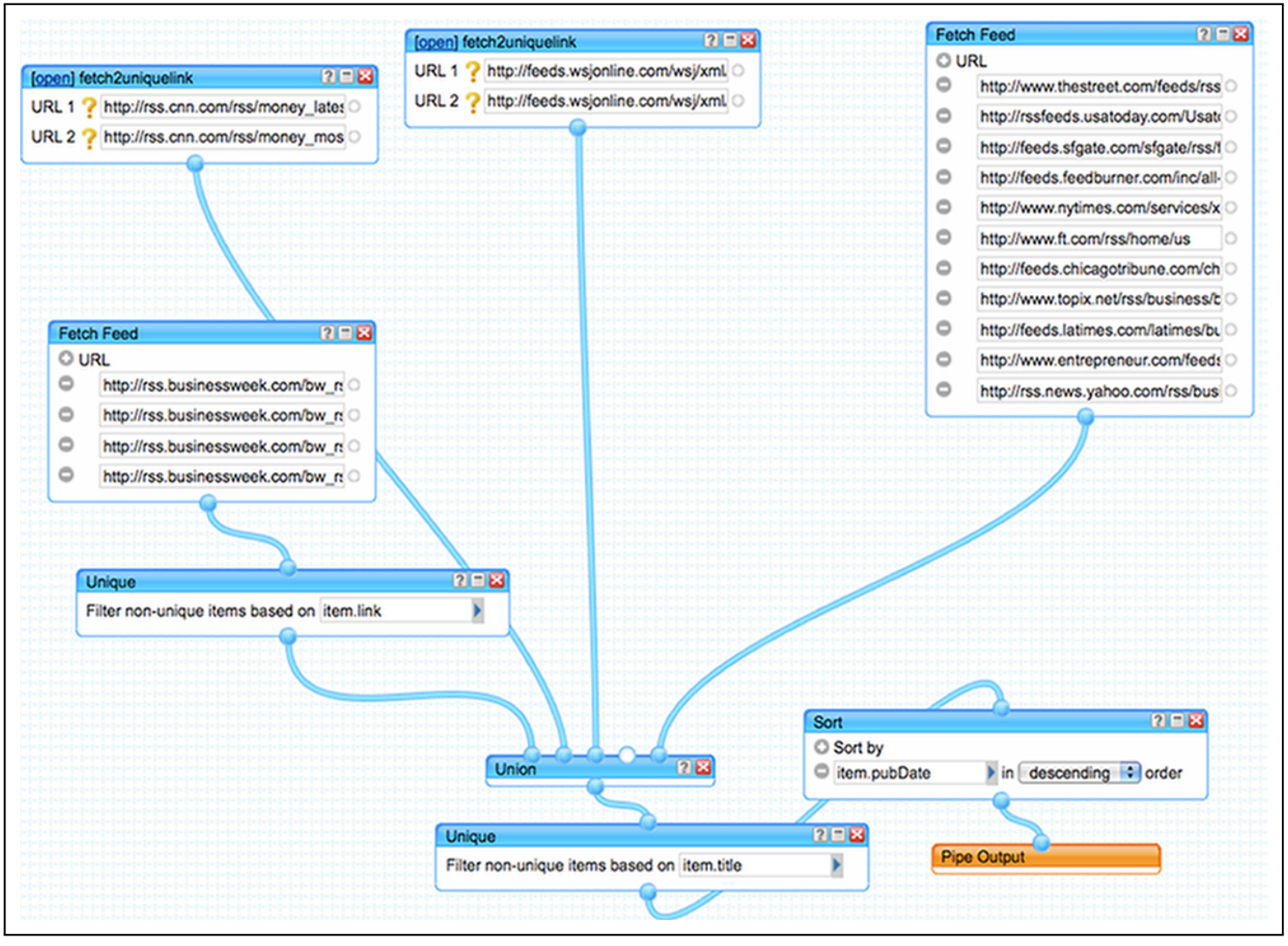
Motivating Example



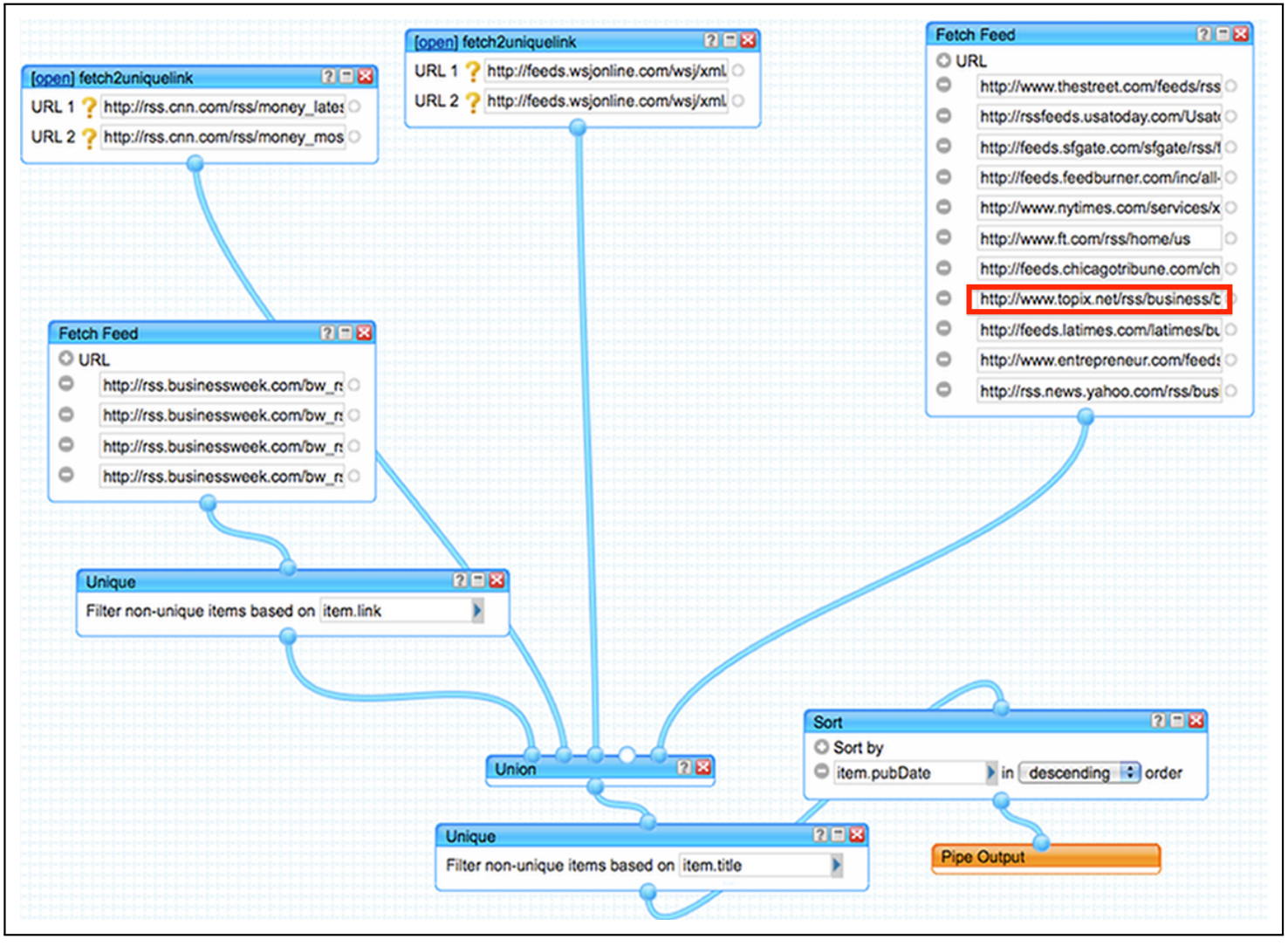
Motivating Example



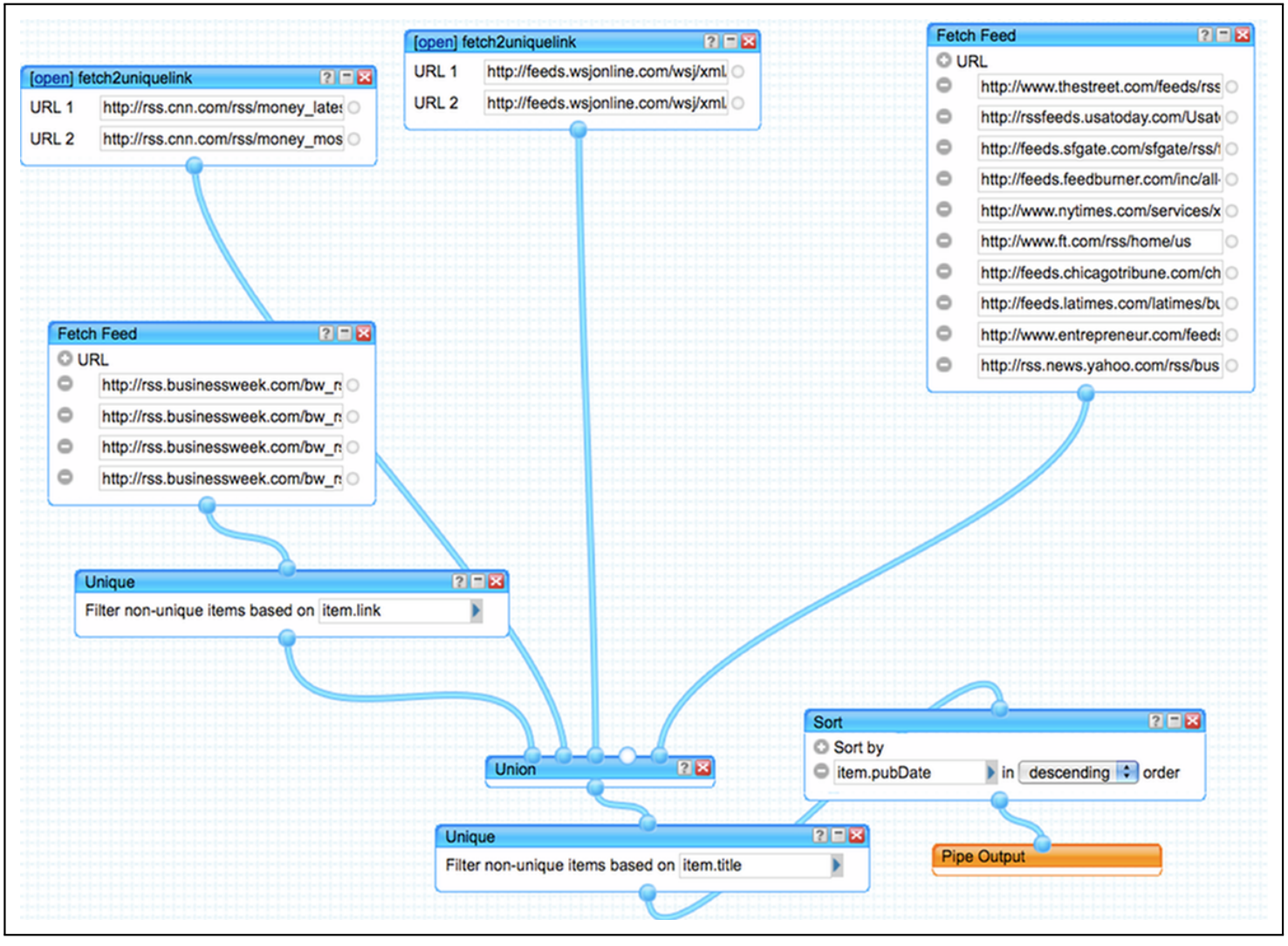
Motivating Example



Motivating Example

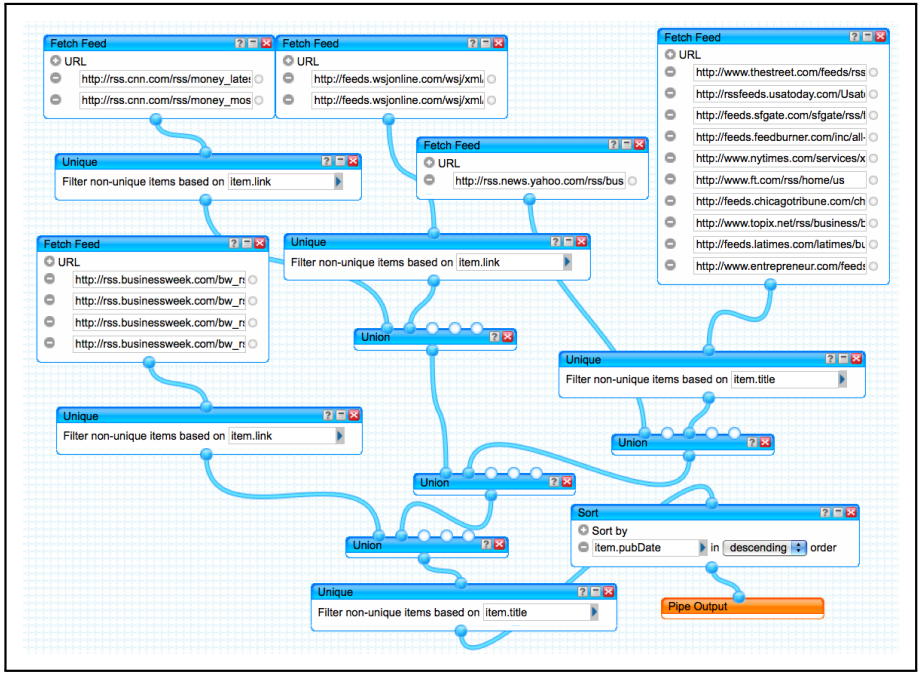


Motivating Example



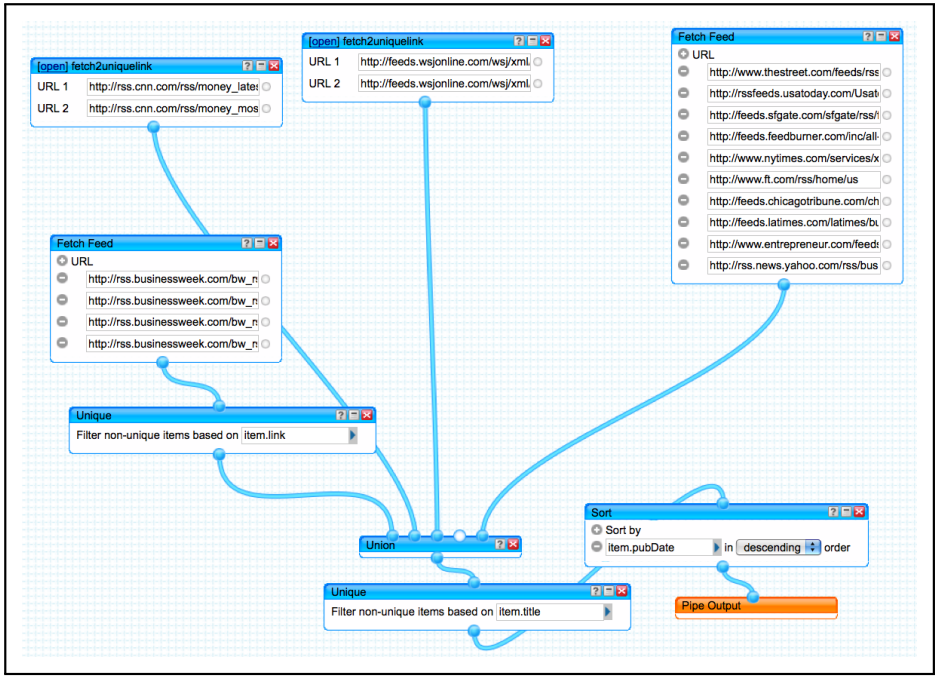
Refactoring in Yahoo! Pipes: Before and After

Before



- Unnecessary modules
- Redundant structures
- Broken data sources

After



- Fewer modules
- More abstract
- All valid data sources

In this talk

- End-user software engineering
- Refactoring for web mashups
- Mashup languages create opportunities to extend refactorings
- Speculate how refactoring adapts to other end user domains

Smells in Mashups

Code Smell

A deficiency in program code that can make it harder to maintain or more error-prone

Smells in Mashups

Code Smell

A deficiency in program code that can make it harder to maintain or more error-prone

We have defined 10 code smells for pipe-like web mashups

Smells in Mashups

Code Smell

A deficiency in program code that can make it harder to maintain or more error-prone

We have defined 10 code smells for pipe-like web mashups

Refactoring

A semantic preserving transformation on a program for the purpose of removing code smells and improving maintainability

Smells in Mashups

Code Smell

A deficiency in program code that can make it harder to maintain or more error-prone

We have defined 10 code smells for pipe-like web mashups

Refactoring

A semantic preserving transformation on a program for the purpose of removing code smells and improving maintainability

We have defined 11 refactorings for pipe-like web mashups

Smell Types

- Laziness – ineffectual or unnecessary components
- Redundancy – duplicate components
- Environmental – external changes cause impact
- Population-Based – deviants from community norms

Smell Types

- **Laziness – ineffectual or unnecessary components**
- **Redundancy – duplicate components**
- Environmental – external changes cause impact
- Population-Based – deviants from community norms

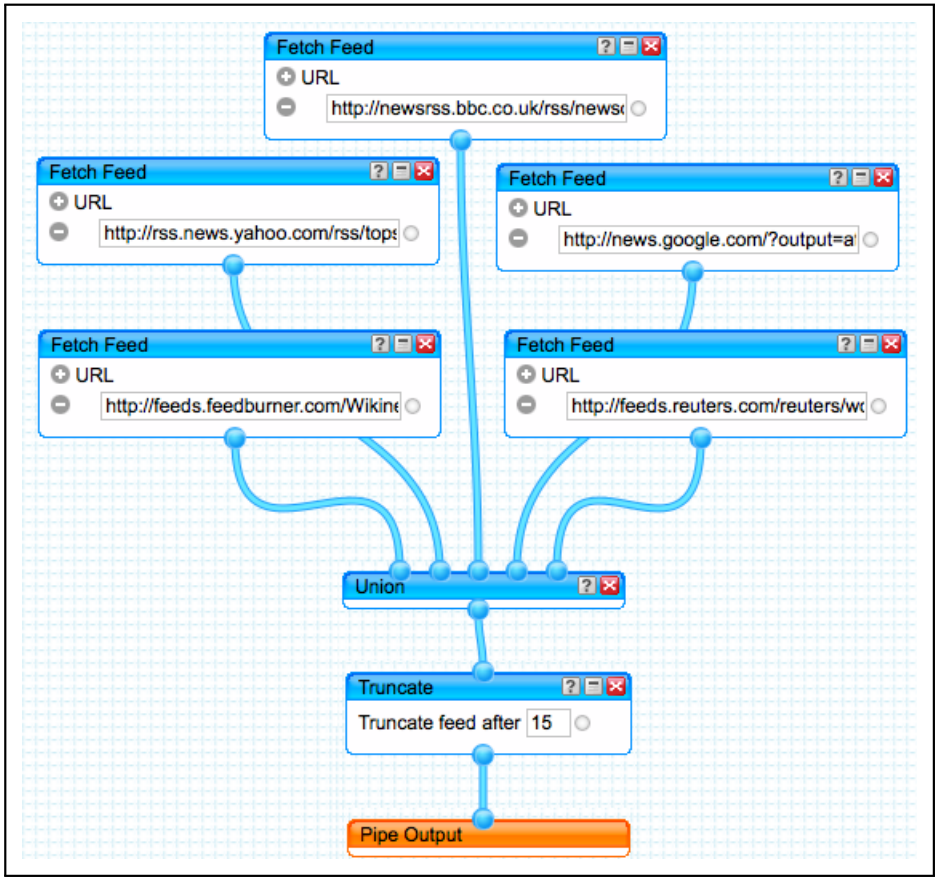
Properties of Languages and Environment

- Flexible and forgiving (e.g., no compiler errors)
- Easy to analyze programs exhaustively

Redundancy and Consolidation

Smell: Duplicate Modules

Similar modules appearing in certain patterns may be redundant and candidate for consolidation.



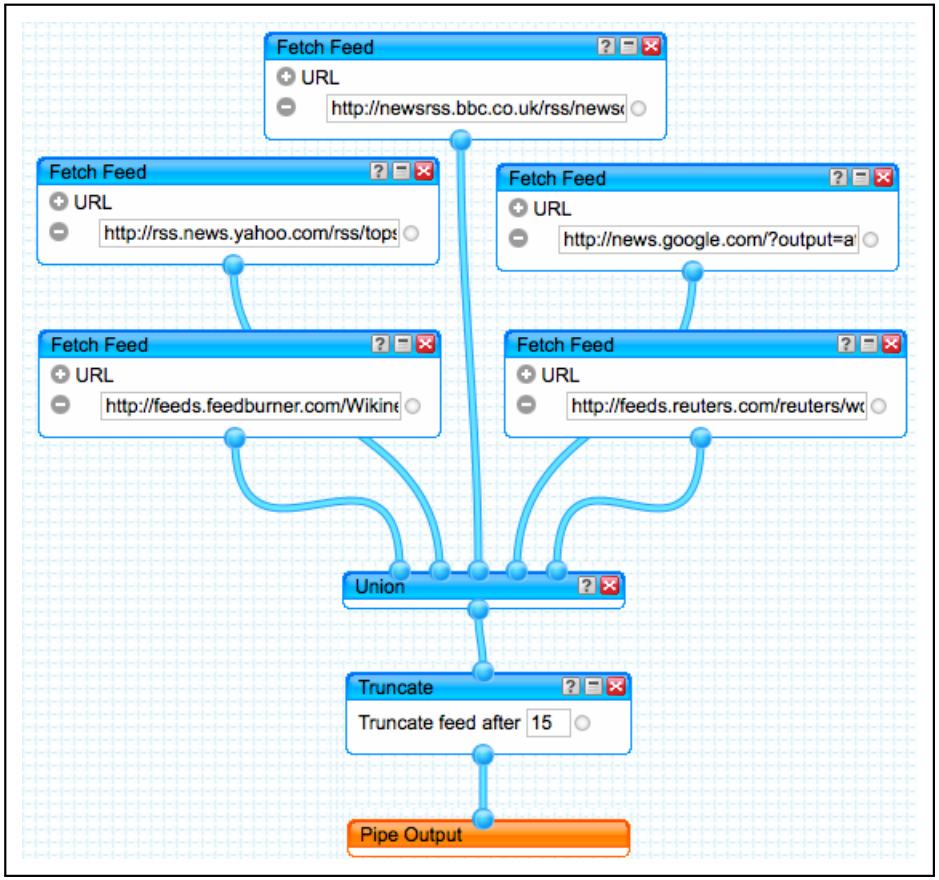
Redundancy and Consolidation

Smell: Duplicate Modules

Similar modules appearing in certain patterns may be redundant and candidate for consolidation.

Definition

$$\exists m, n \in \mathcal{M} \mid m \neq n \wedge$$
$$m.name = n.name \wedge$$
$$gen(m) \wedge gen(n) \wedge$$
$$connected\_to\_union(m, n)$$



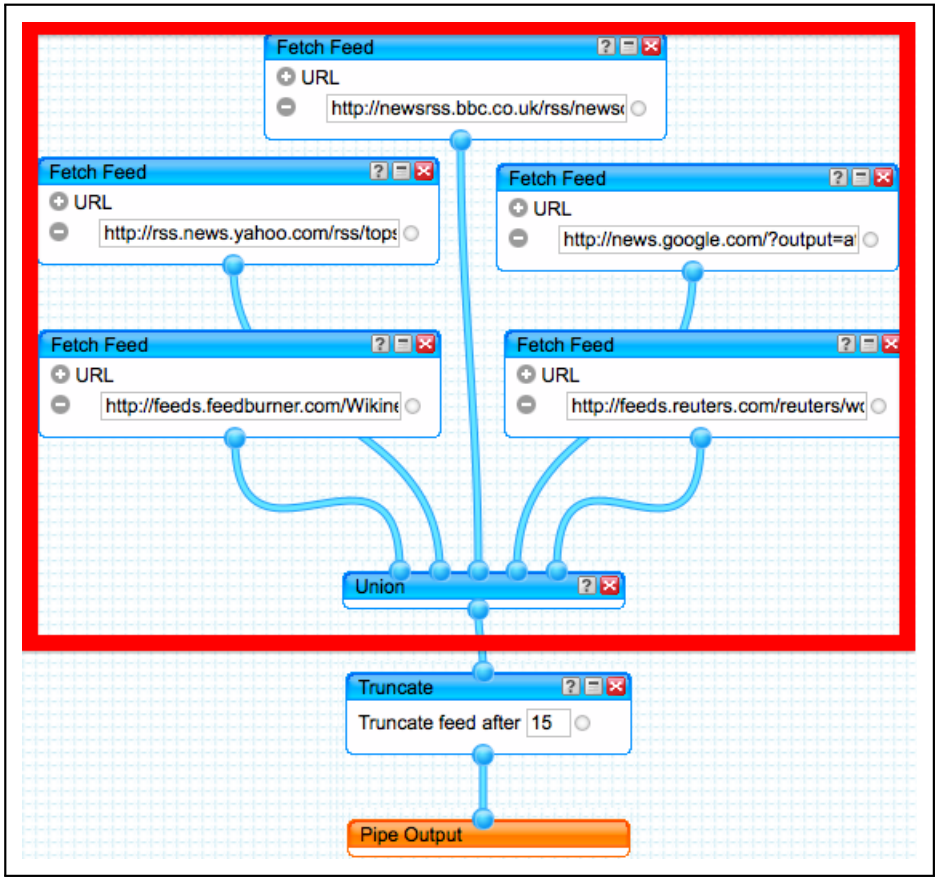
Redundancy and Consolidation

Smell: Duplicate Modules

Similar modules appearing in certain patterns may be redundant and candidate for consolidation.

Definition

$$\exists m, n \in \mathcal{M} \mid m \neq n \wedge$$
$$m.name = n.name \wedge$$
$$gen(m) \wedge gen(n) \wedge$$
$$connected\_to\_union(m, n)$$



Redundancy and Consolidation

Smell: Duplicate Modules

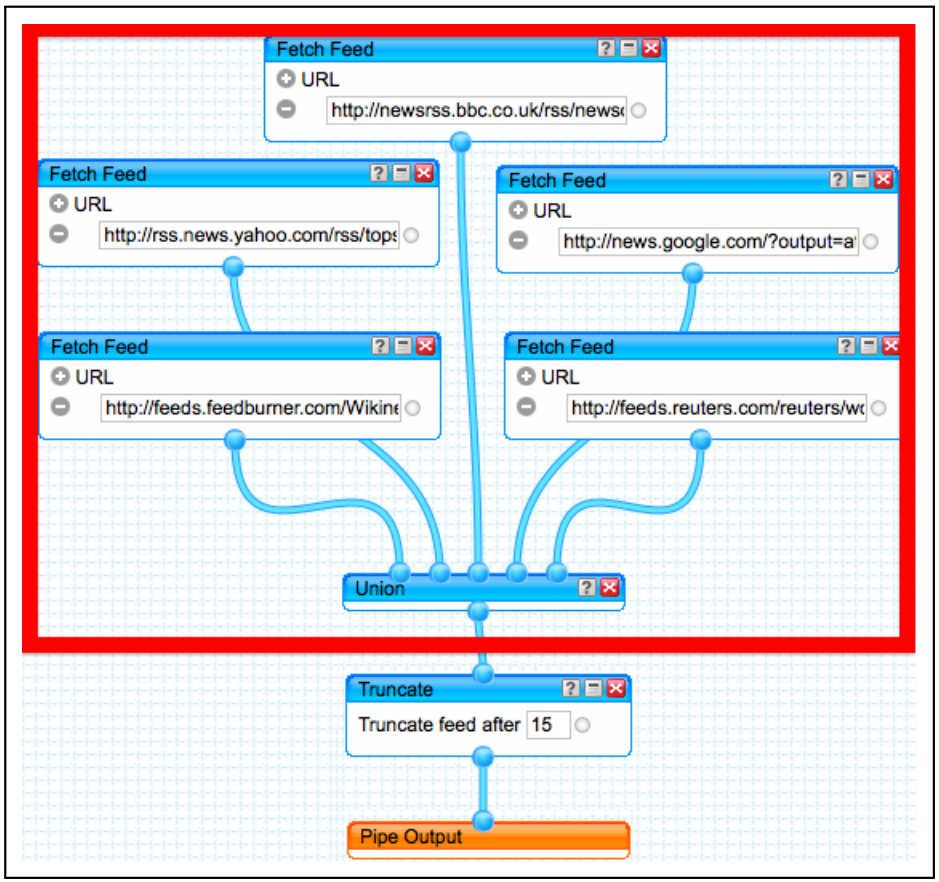
Similar modules appearing in certain patterns may be redundant and candidate for consolidation.

Definition

$$\exists m, n \in \mathcal{M} \mid m \neq n \wedge$$
$$m.name = n.name \wedge$$
$$gen(m) \wedge gen(n) \wedge$$
$$connected\_to\_union(m, n)$$

Why is this Smelly?

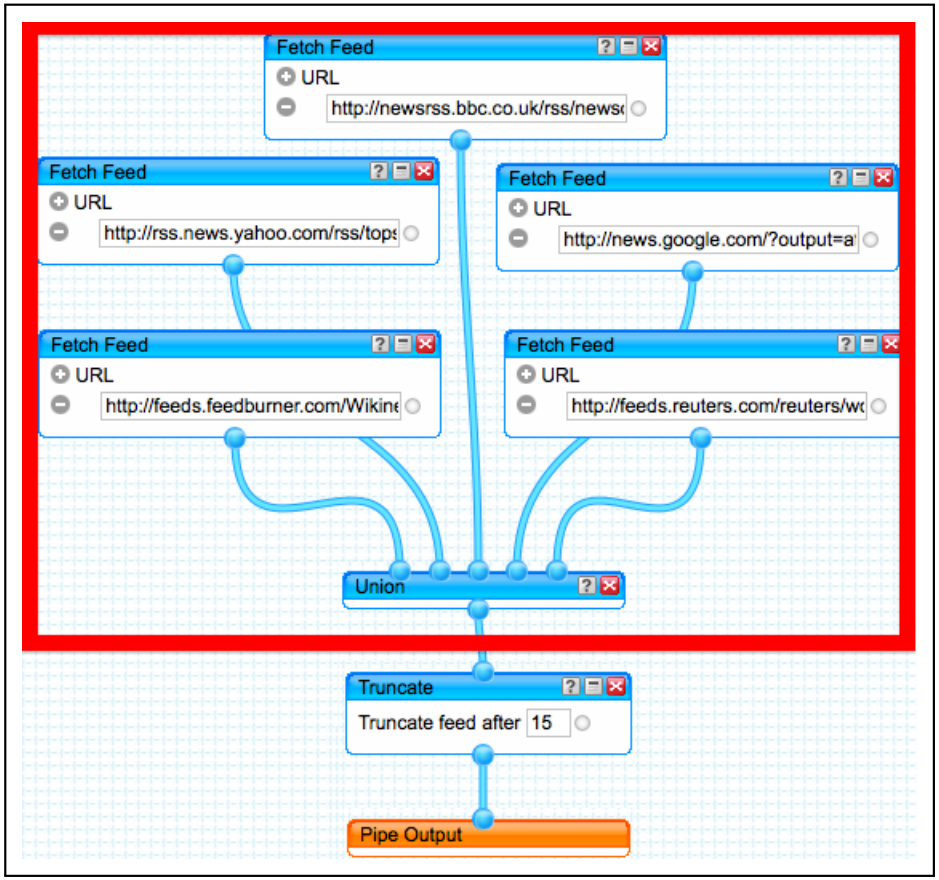
Adds unnecessary complexity



Redundancy and Consolidation

Refactor: Collapse Duplicate Paths

Paths that are aggregated using the same union module can often be consolidated into a single path



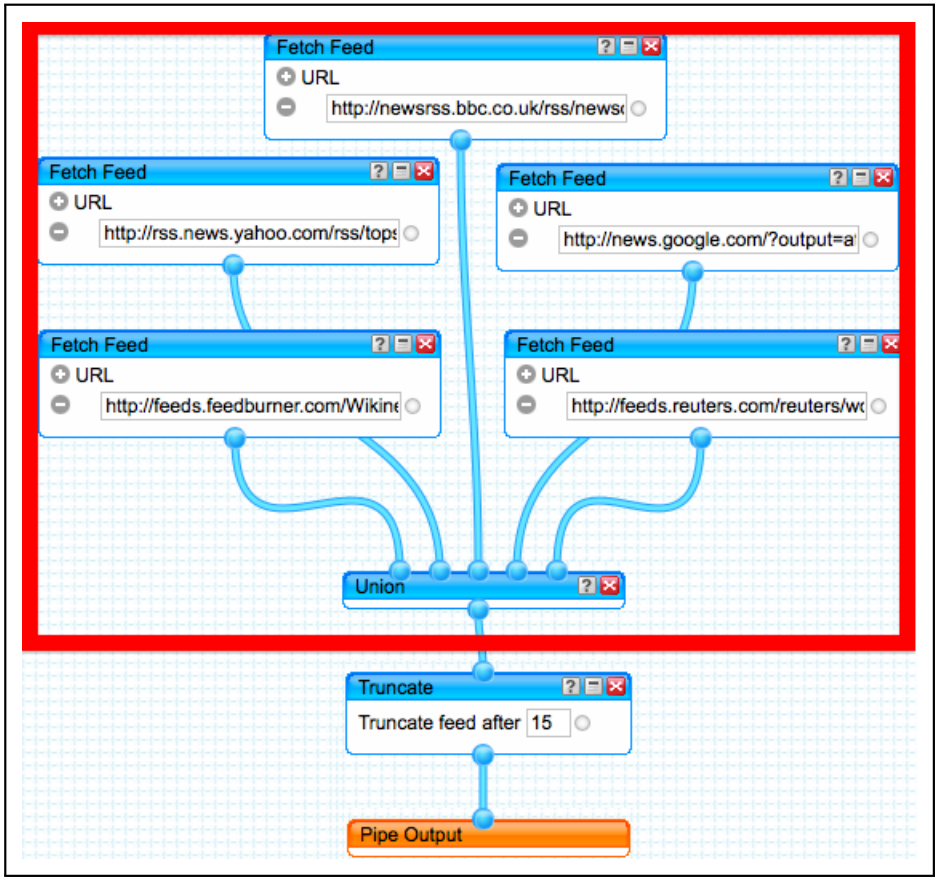
Redundancy and Consolidation

Refactor: Collapse Duplicate Paths

Paths that are aggregated using the same union module can often be consolidated into a single path

Definition

P_{before}	Joined generators
Params	$Pipe = (\mathcal{M}, \mathcal{W}, \mathcal{F}, owner)$, joined modules m, n
Transf.	$\forall f \in n.\mathcal{F}$ move f to m
	$\exists w \in \mathcal{W} \mid out\_wire(n, w)$ remove w
	remove n
P_{after}	$n, w \notin Pipe$
	$m'.\mathcal{F} = n.\mathcal{F} \cup m.\mathcal{F}$



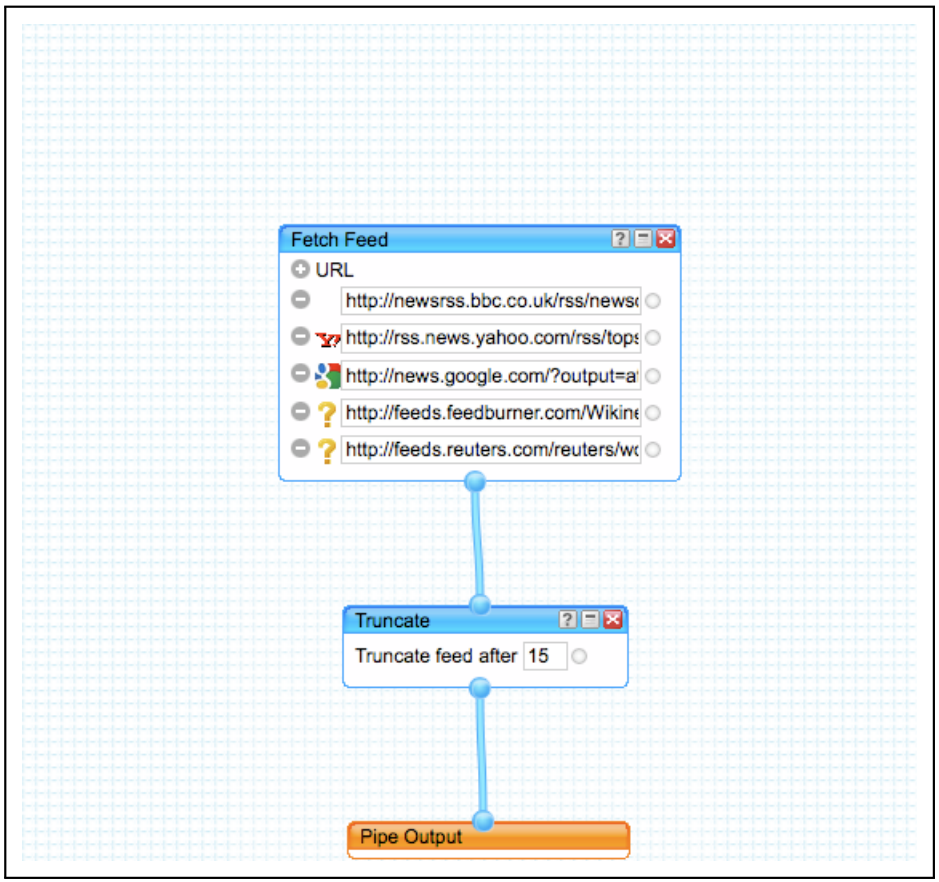
Redundancy and Consolidation

Refactor: Collapse Duplicate Paths

Paths that are aggregated using the same union module can often be consolidated into a single path

Definition

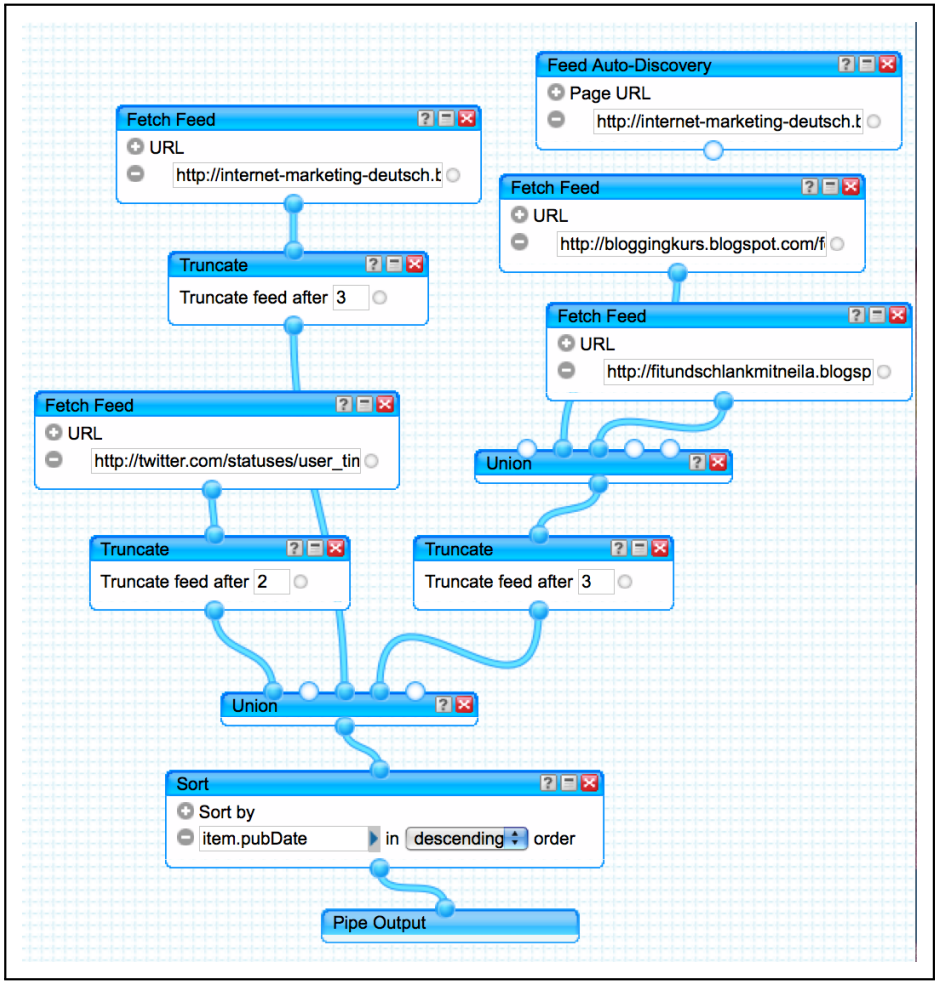
P_{before}	Joined generators
Params	$Pipe = (\mathcal{M}, \mathcal{W}, \mathcal{F}, owner)$, joined modules m, n
Transf.	$\forall f \in n.\mathcal{F}$ move f to m $\exists w \in \mathcal{W} \mid out\_wire(n, w)$ remove w remove n
P_{after}	$n, w \notin Pipe$ $m'.\mathcal{F} = n.\mathcal{F} \cup m.\mathcal{F}$



Laziness and Reduction

Smell: Unnecessary Module

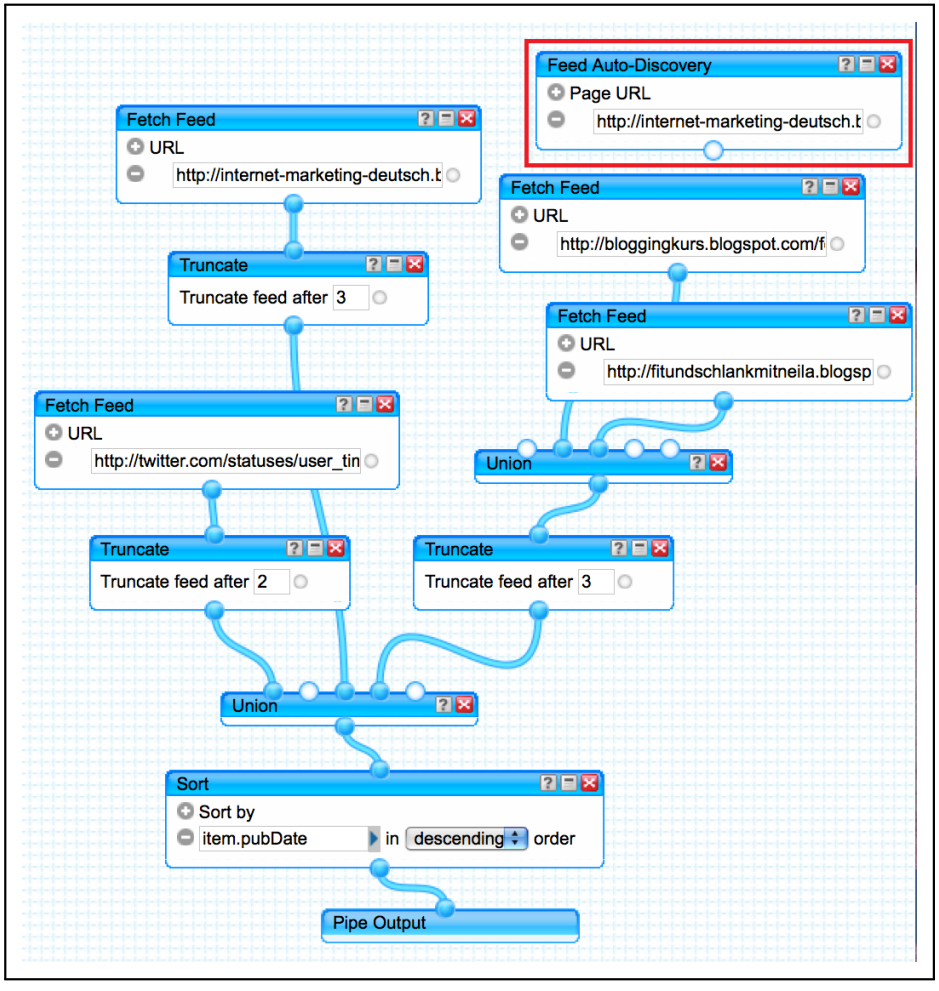
A module whose execution does not affect the pipe's output



Laziness and Reduction

Smell: Unnecessary Module

A module whose execution does not affect the pipe's output



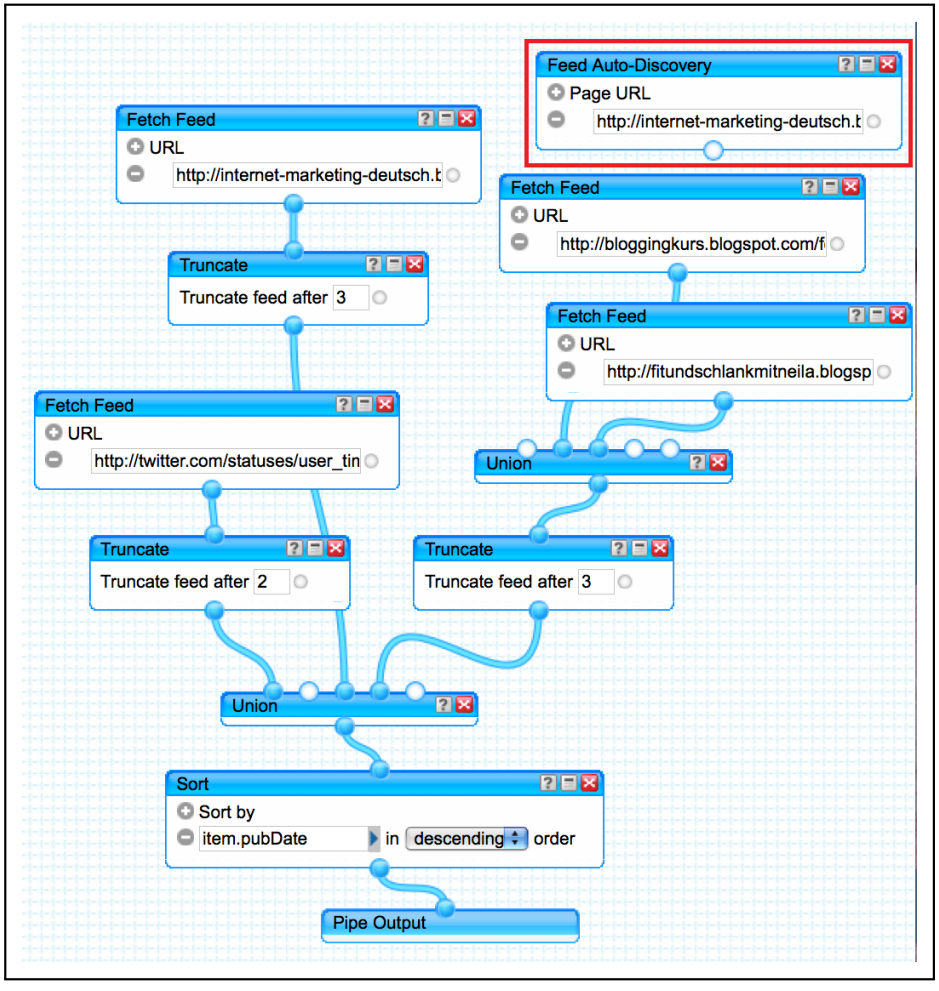
Laziness and Reduction

Smell: Unnecessary Module

A module whose execution does not affect the pipe's output

Why is this Smelly?

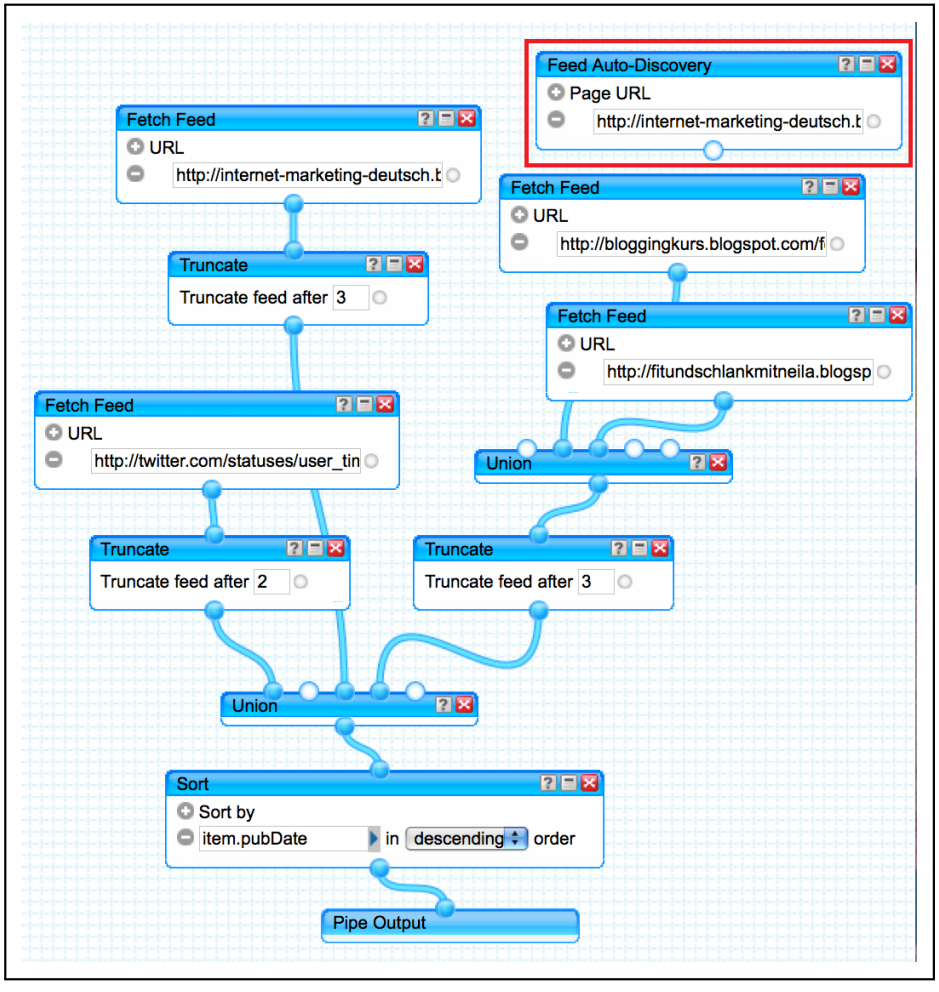
It's dead code



Laziness and Reduction

Refactor: Remove Non-Contributing Module

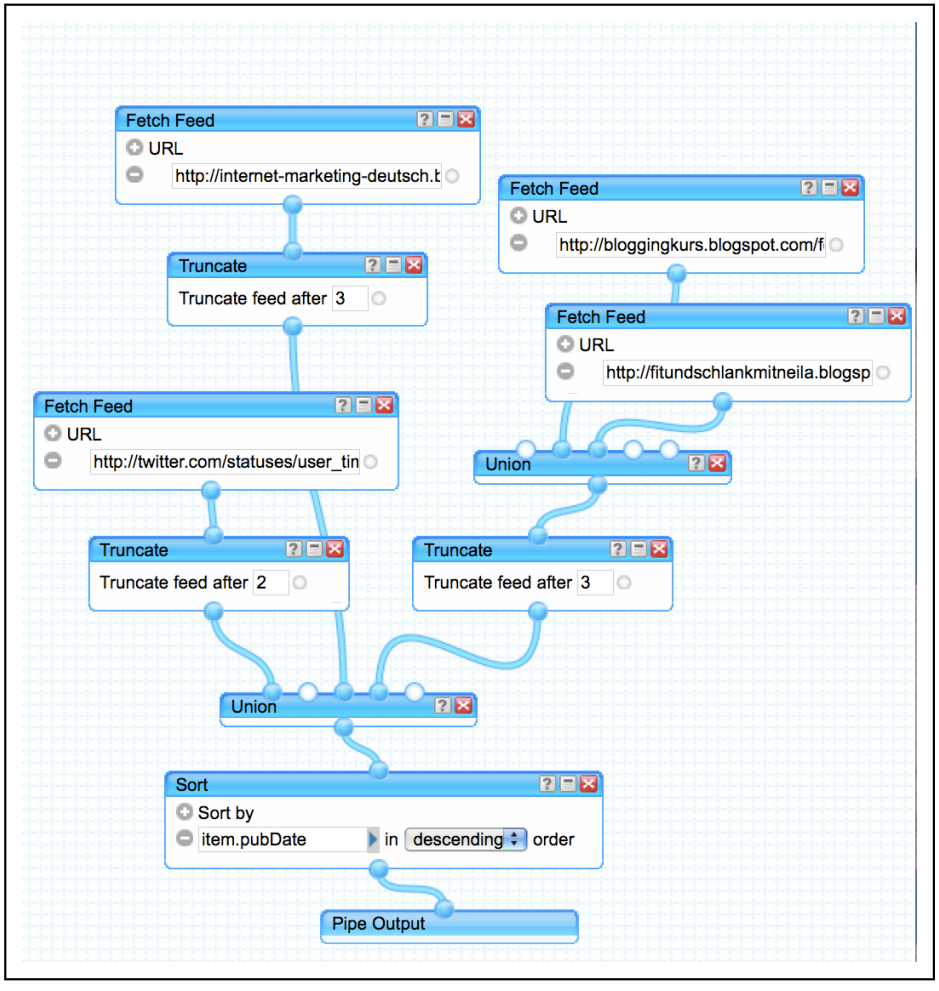
Modules that do not reach the output and their attaching wires can be removed.



Laziness and Reduction

Refactor: Remove Non-Contributing Module

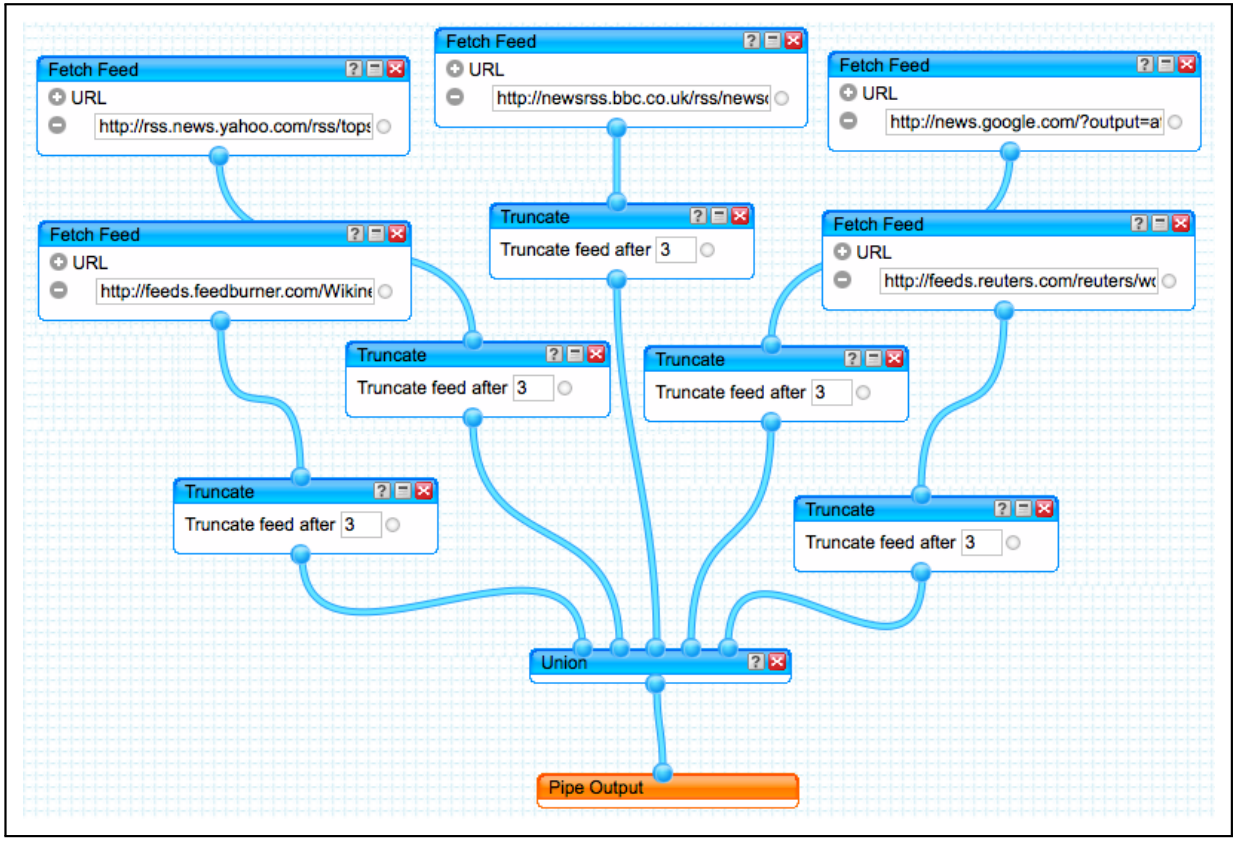
Modules that do not reach the output and their attaching wires can be removed.



Redundancy and Abstraction

Smell: Duplicated String

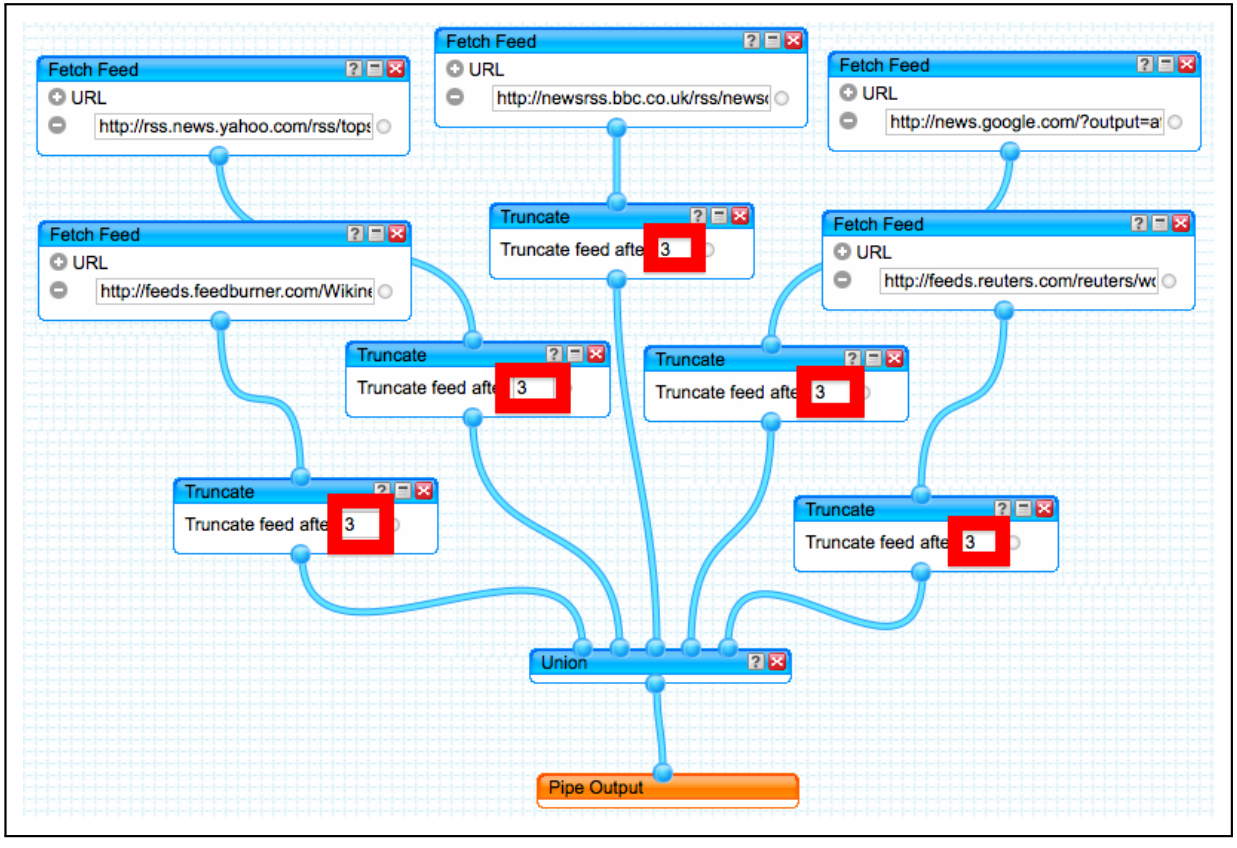
A constant string that is used in at least *n* wireable fields in at least *m* modules.



Redundancy and Abstraction

Smell: Duplicated String

A constant string that is used in at least *n* wireable fields in at least *m* modules.



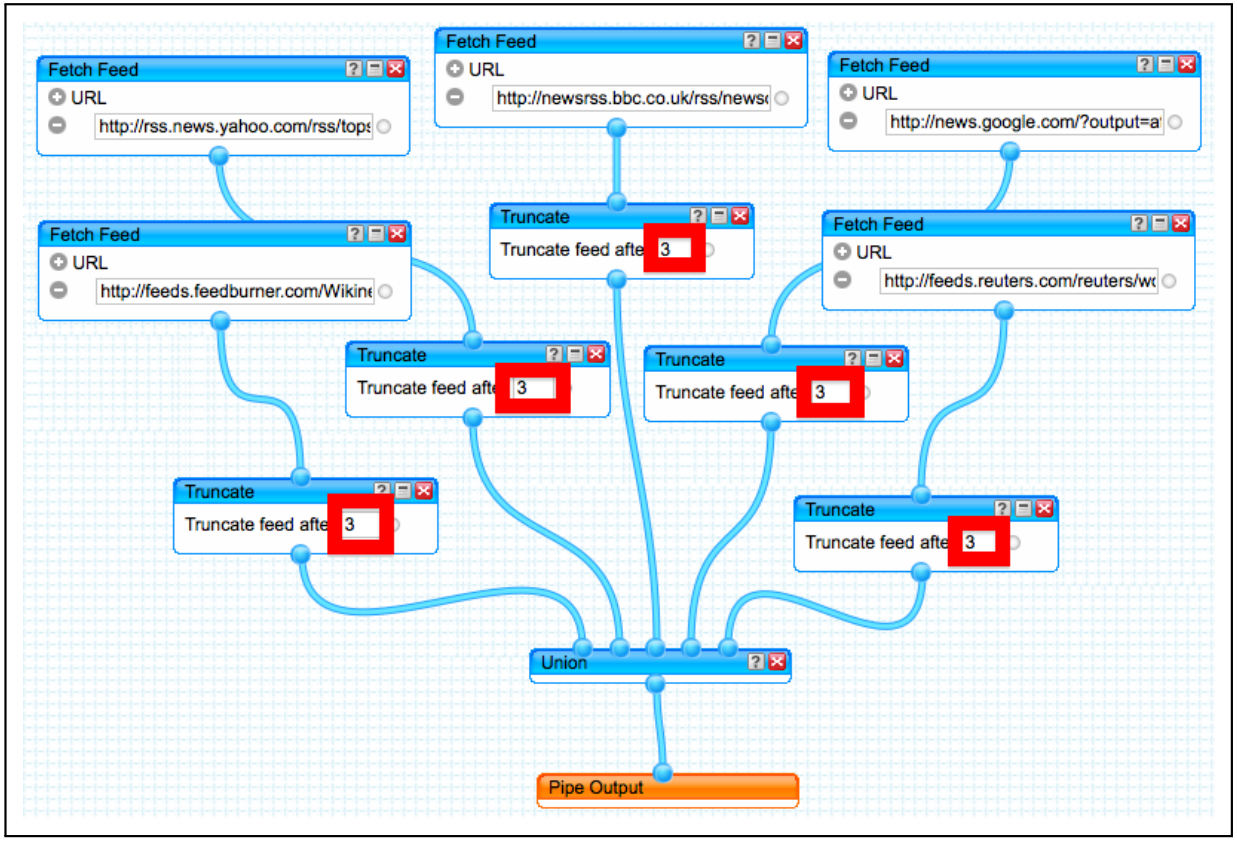
Redundancy and Abstraction

Smell: Duplicated String

A constant string that is used in at least *n* wireable fields in at least *m* modules.

Why is this Smelly?

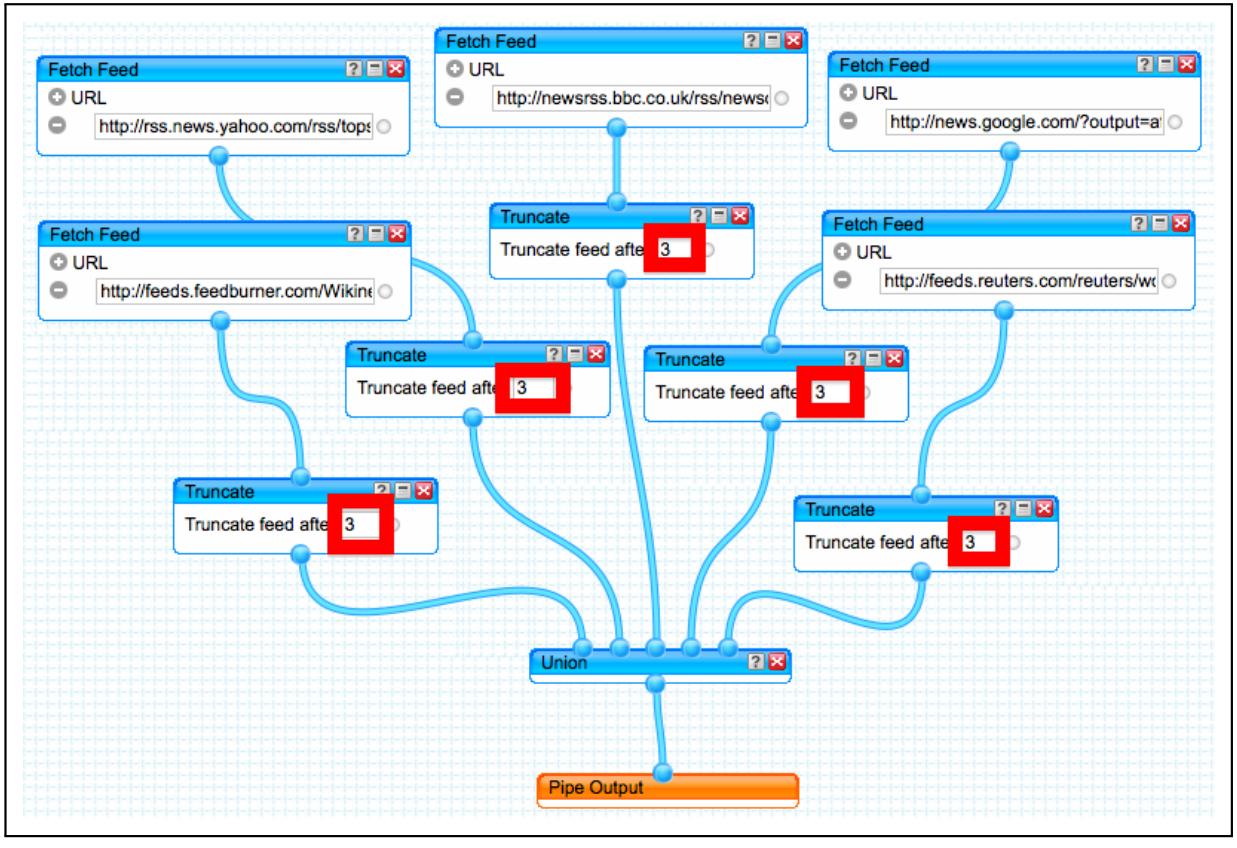
Hard to maintain; missed opportunity for abstraction



Redundancy and Abstraction

Refactor: Pull Up Module

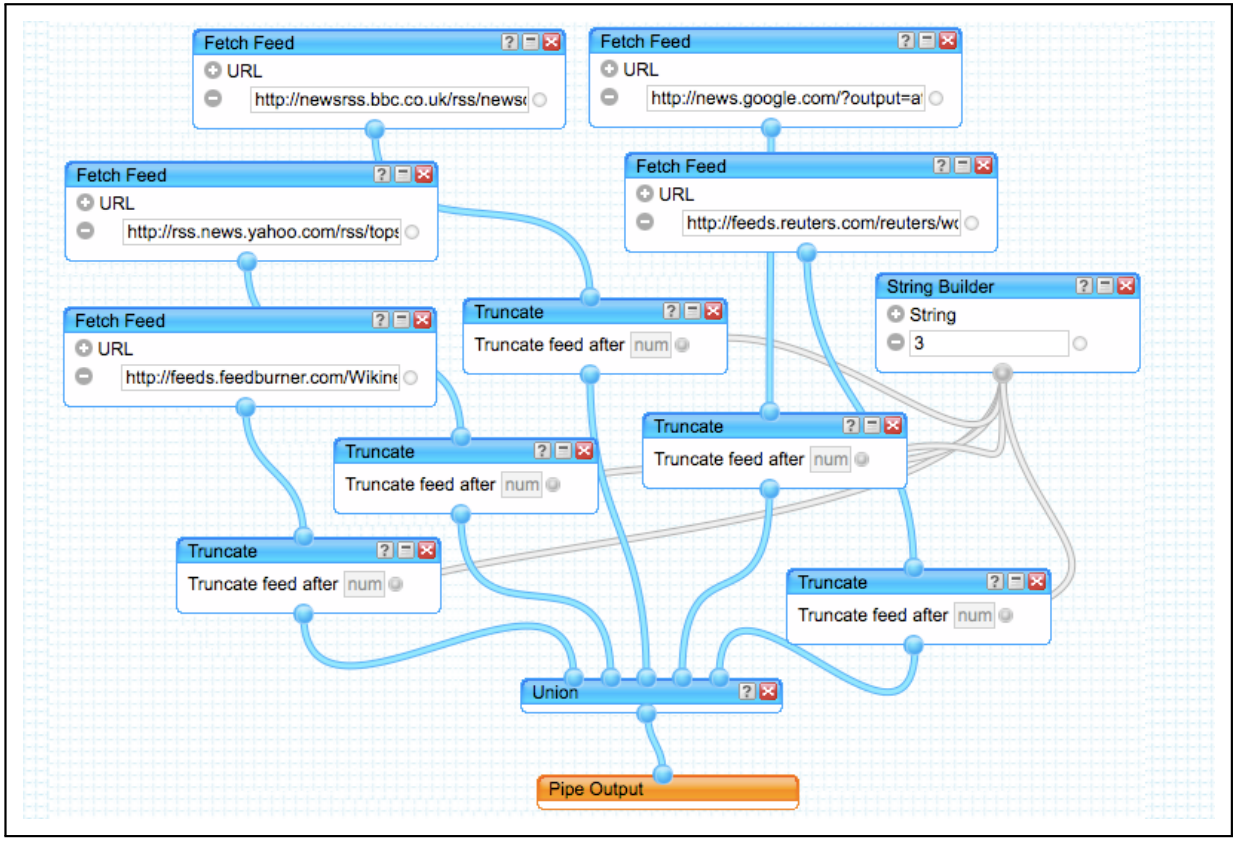
Extracts duplicate strings into a newly created module and provides string values via wires to the previous owners of the strings



Redundancy and Abstraction

Refactor: Pull Up Module

Extracts duplicate strings into a newly created module and provides string values via wires to the previous owners of the strings



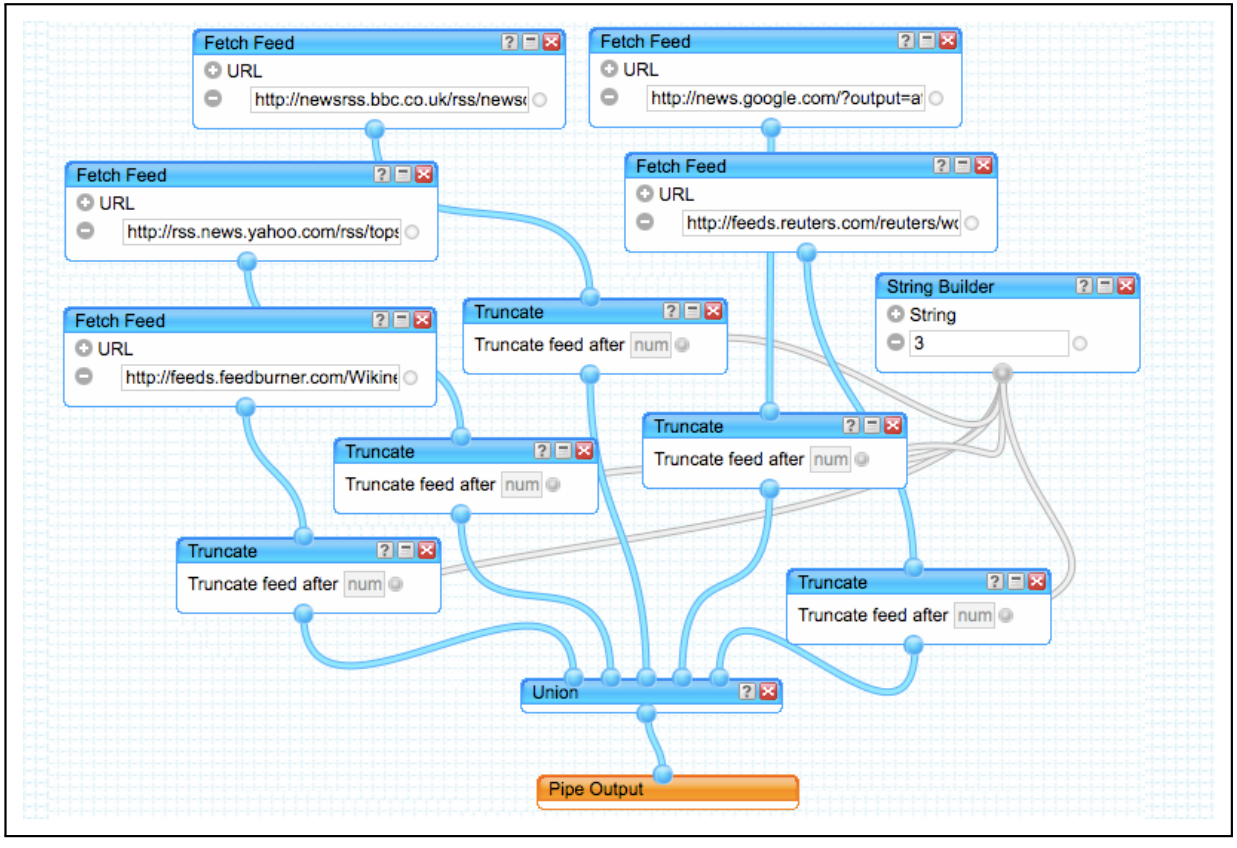
Redundancy and Abstraction

Refactor: Pull Up Module

Extracts duplicate strings into a newly created module and provides string values via wires to the previous owners of the strings

Inspiration

Pull Up Method refactoring



Smell Types

- Laziness – ineffectual or unnecessary components
- Redundancy – duplicate components
- **Environmental – external changes cause impact**
- Population-Based – deviants from community norms

Properties of Internet Resources:*

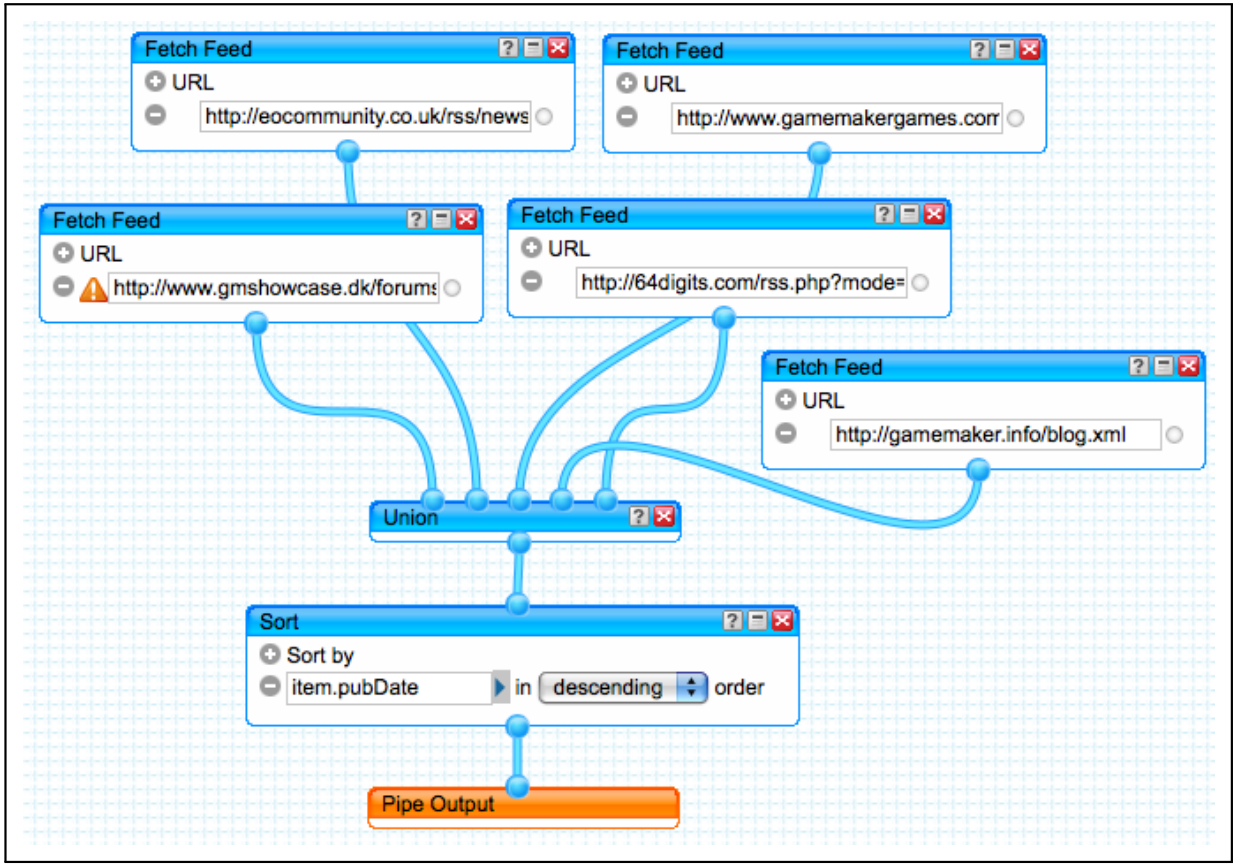
- Independently created and managed
- May change structure or format without notice

* Credit to Mary Shaw

Environmental and Deprecation

Smell: Invalid Source

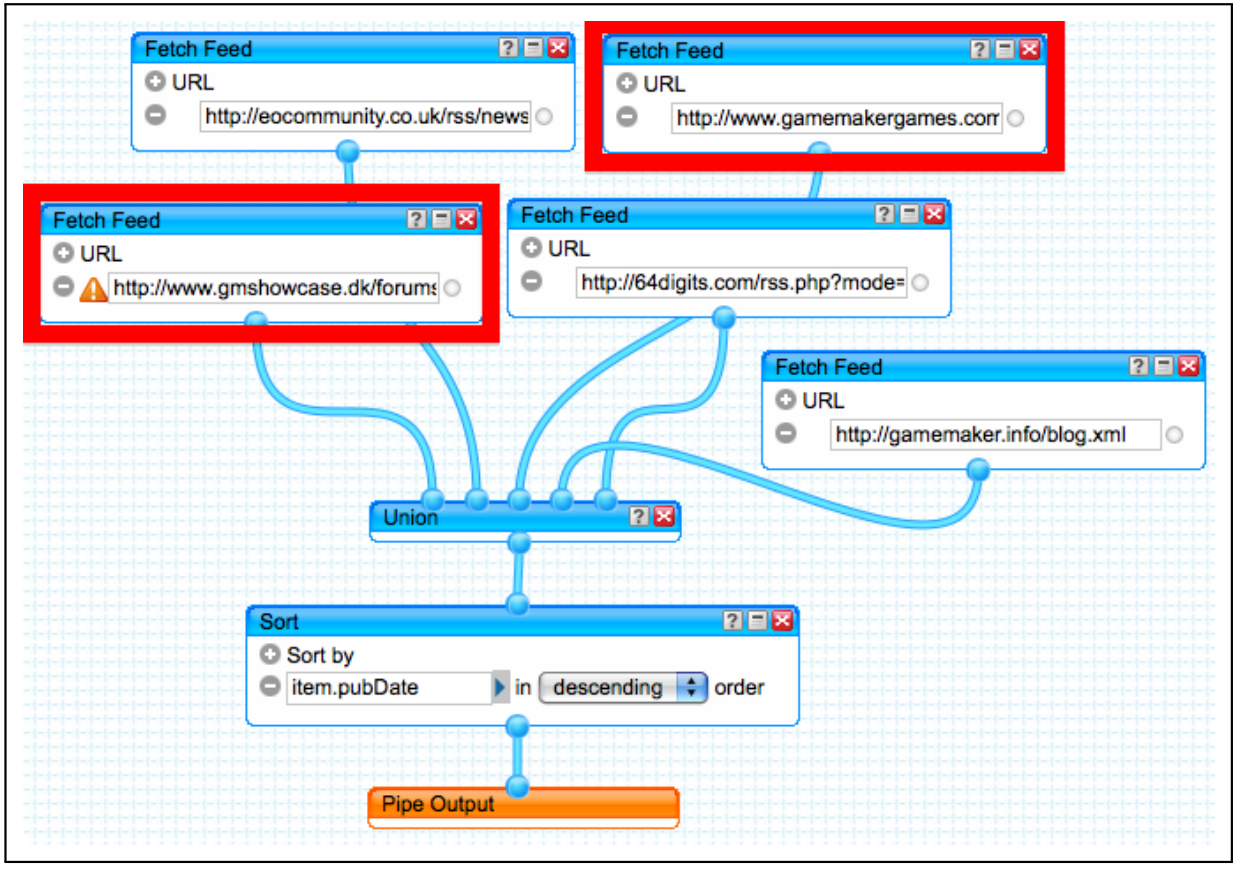
An external data source is invalid if *n* consecutive attempts to retrieve data from it report errors



Environmental and Deprecation

Smell: Invalid Source

An external data source is invalid if *n* consecutive attempts to retrieve data from it report errors



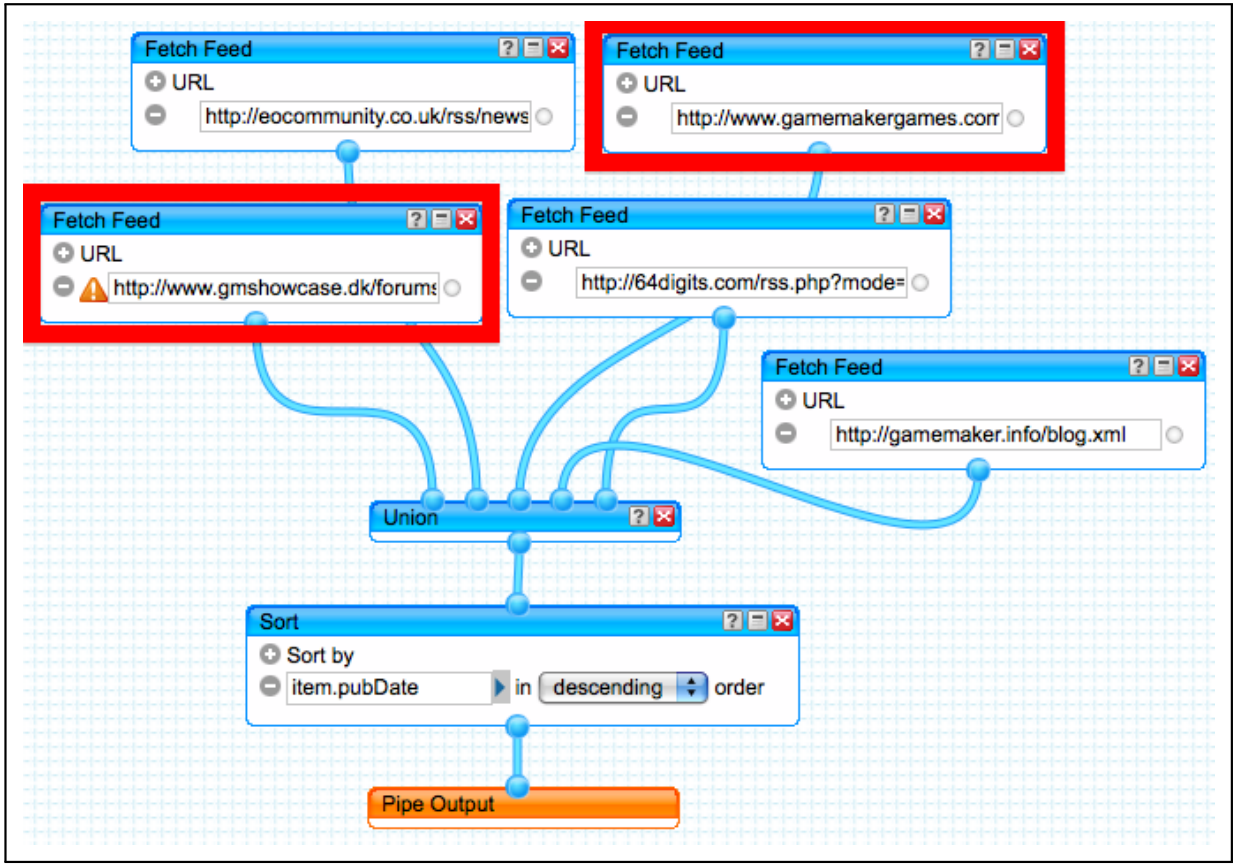
Environmental and Deprecation

Smell: Invalid Source

An external data source is invalid if *n* consecutive attempts to retrieve data from it report errors

Why is this Smelly?

Bloats the code; introduces nondeterminism



Environmental and Deprecation

Smell: Invalid Source

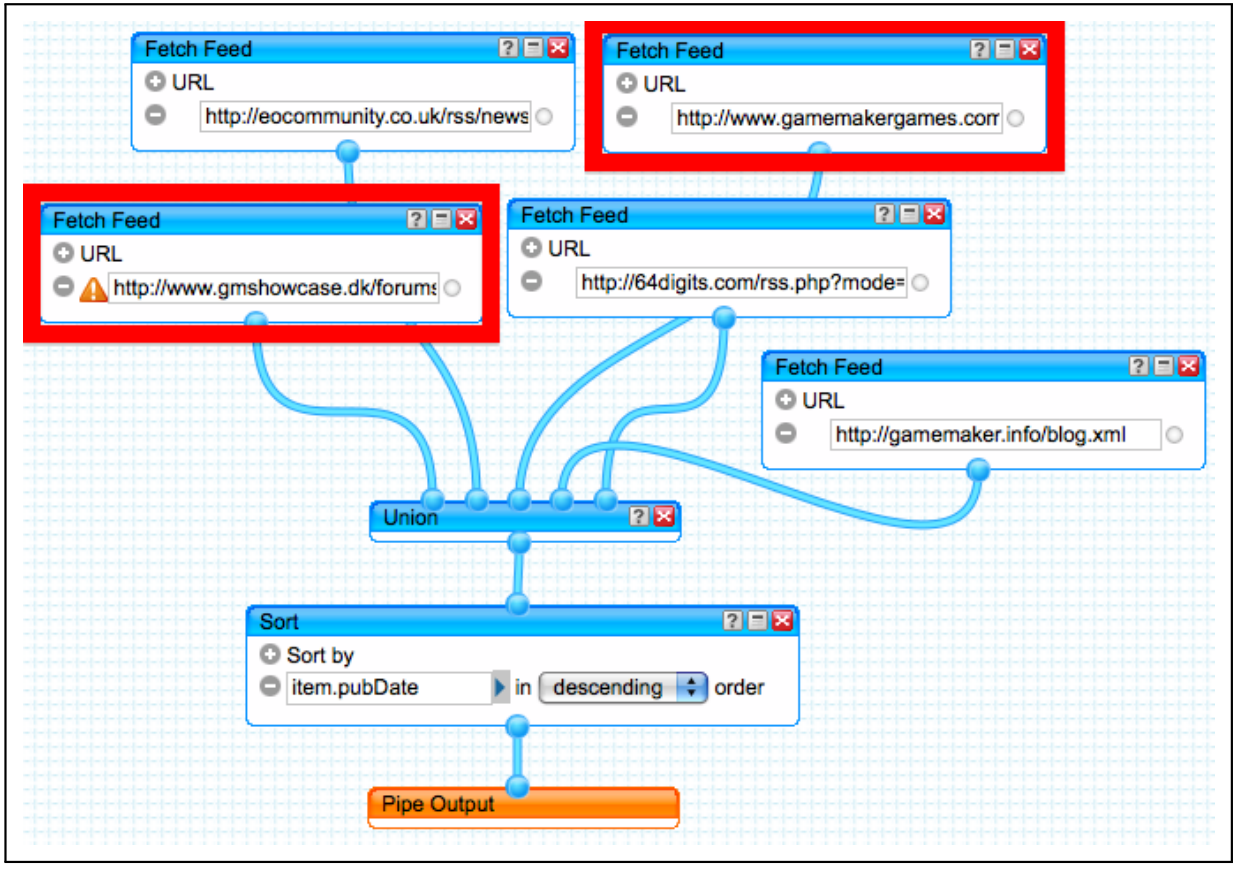
An external data source is invalid if *n* consecutive attempts to retrieve data from it report errors

Why is this Smelly?

Bloats the code; introduces nondeterminism

Insight

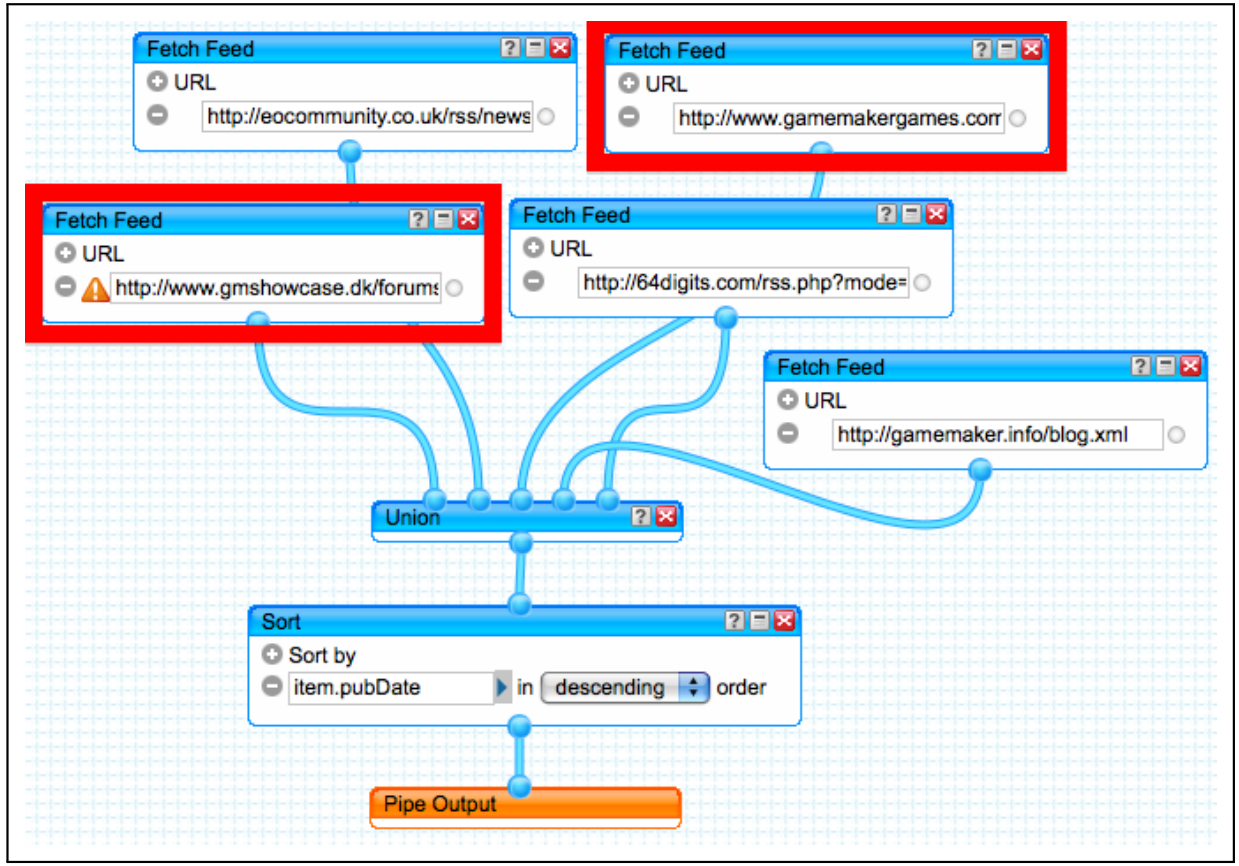
Internet resources are subject to change without notice



Environmental and Deprecation

Refactor: Remove Deprecated Sources

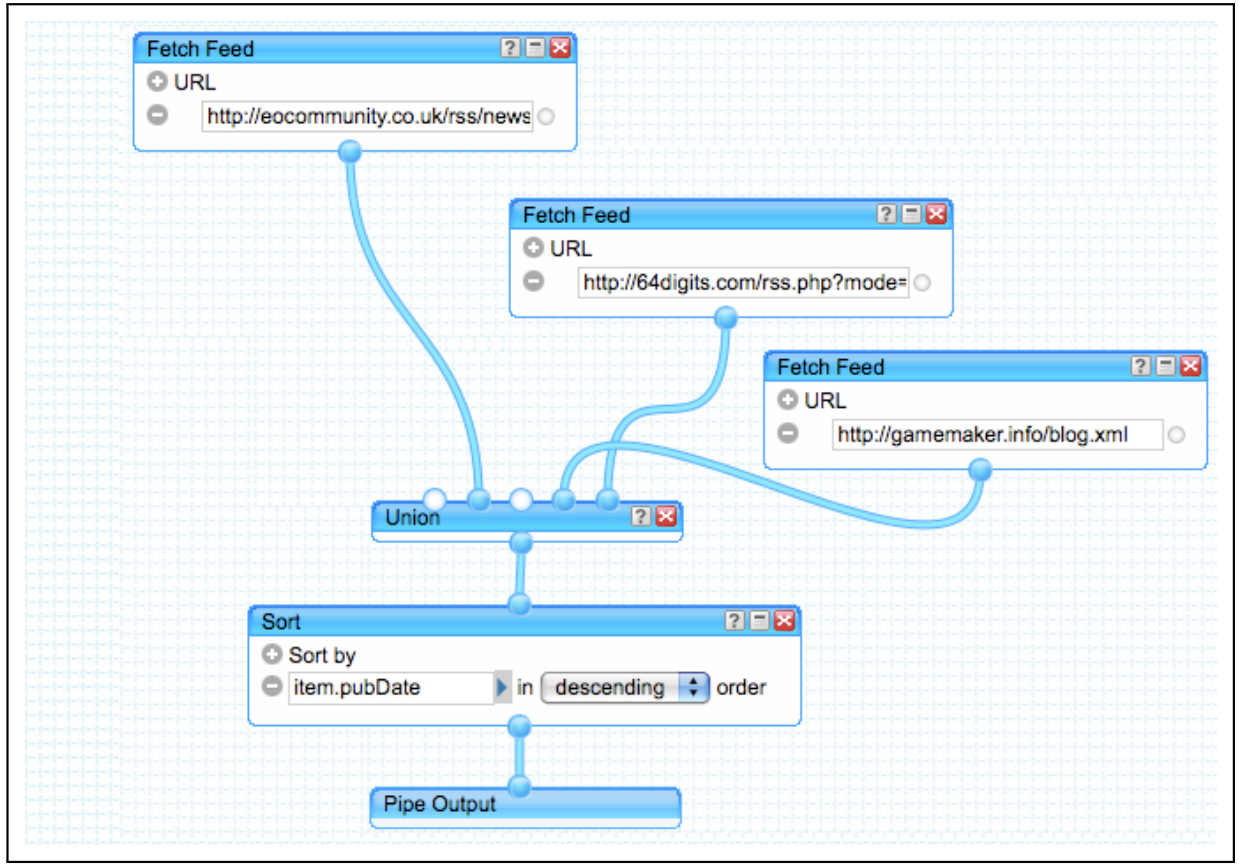
Remove all references to invalid external data sources



Environmental and Deprecation

Refactor: Remove Deprecated Sources

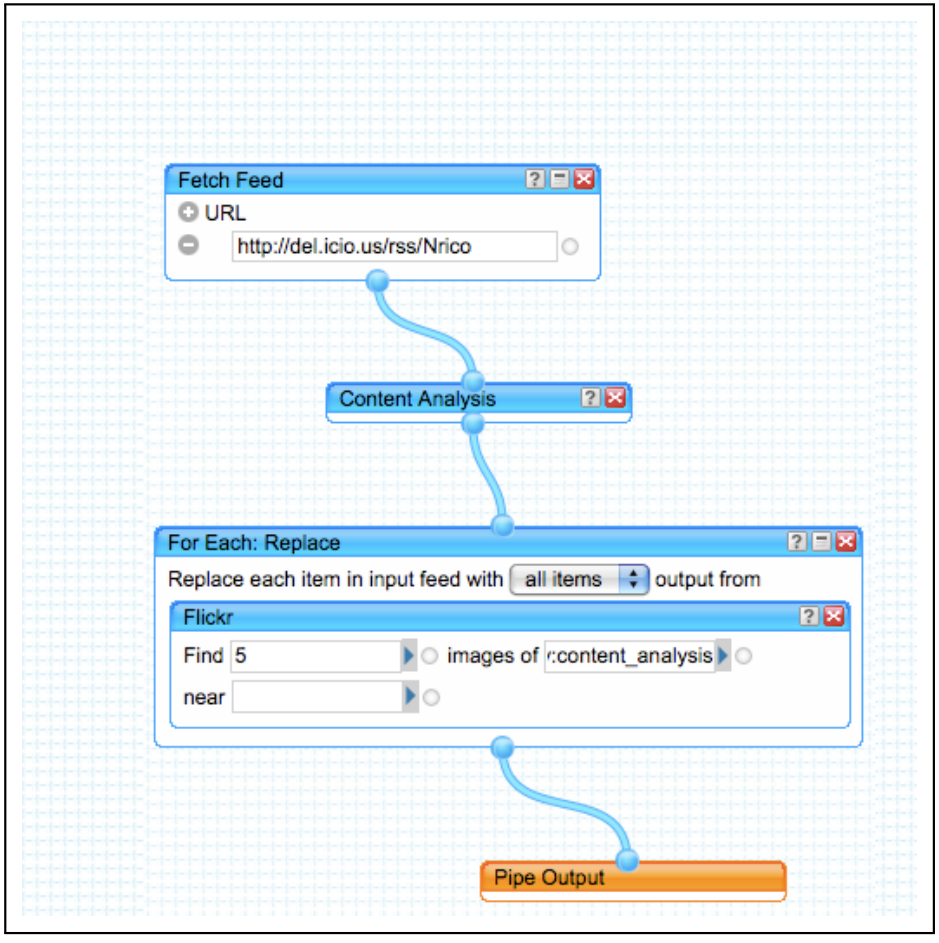
Remove all references to invalid external data sources



Environmental and Deprecation

Smell: Deprecated Module

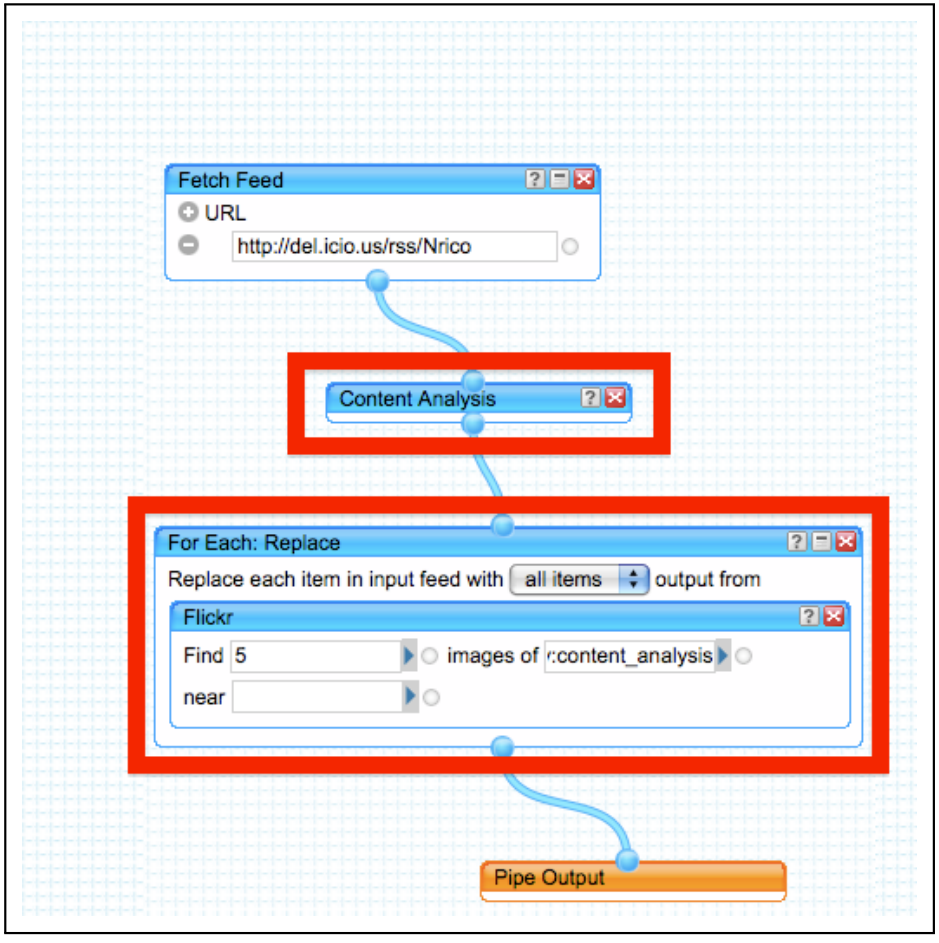
A module that is no longer supported by the environment



Environmental and Deprecation

Smell: Deprecated Module

A module that is no longer supported by the environment



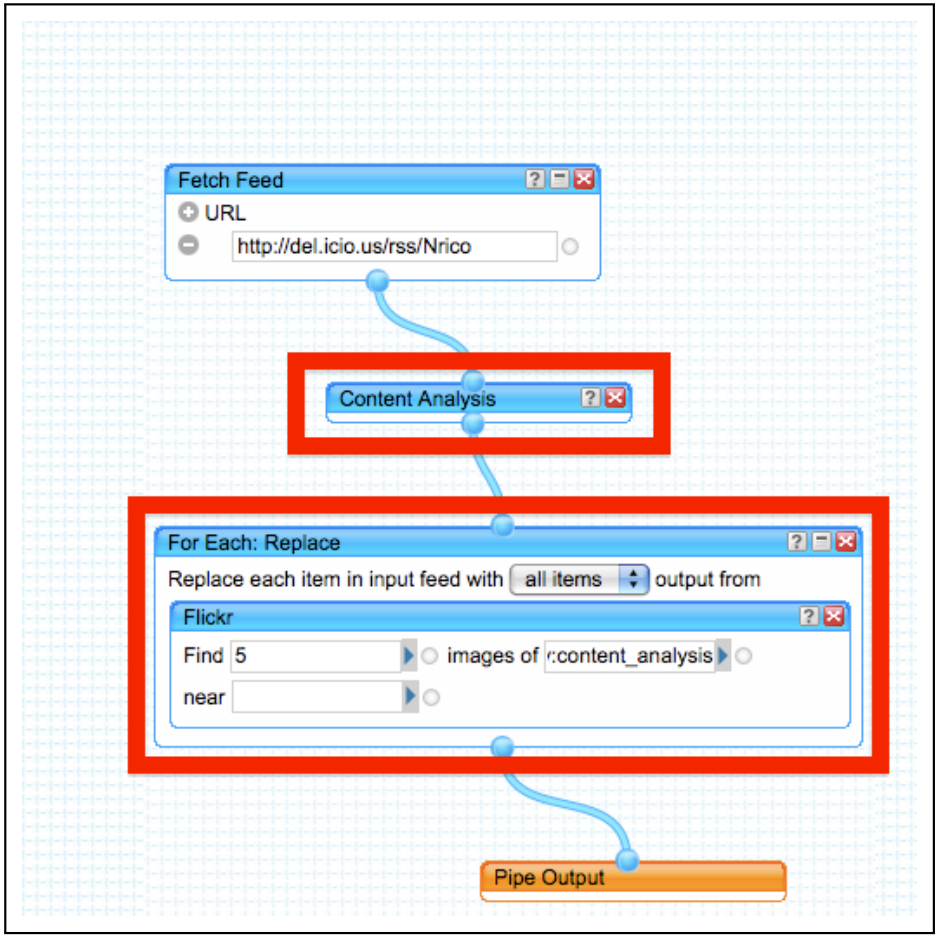
Environmental and Deprecation

Smell: Deprecated Module

A module that is no longer supported by the environment

Why is this Smelly?

Module may cause unexpected or faulty behavior



Environmental and Deprecation

Smell: Deprecated Module

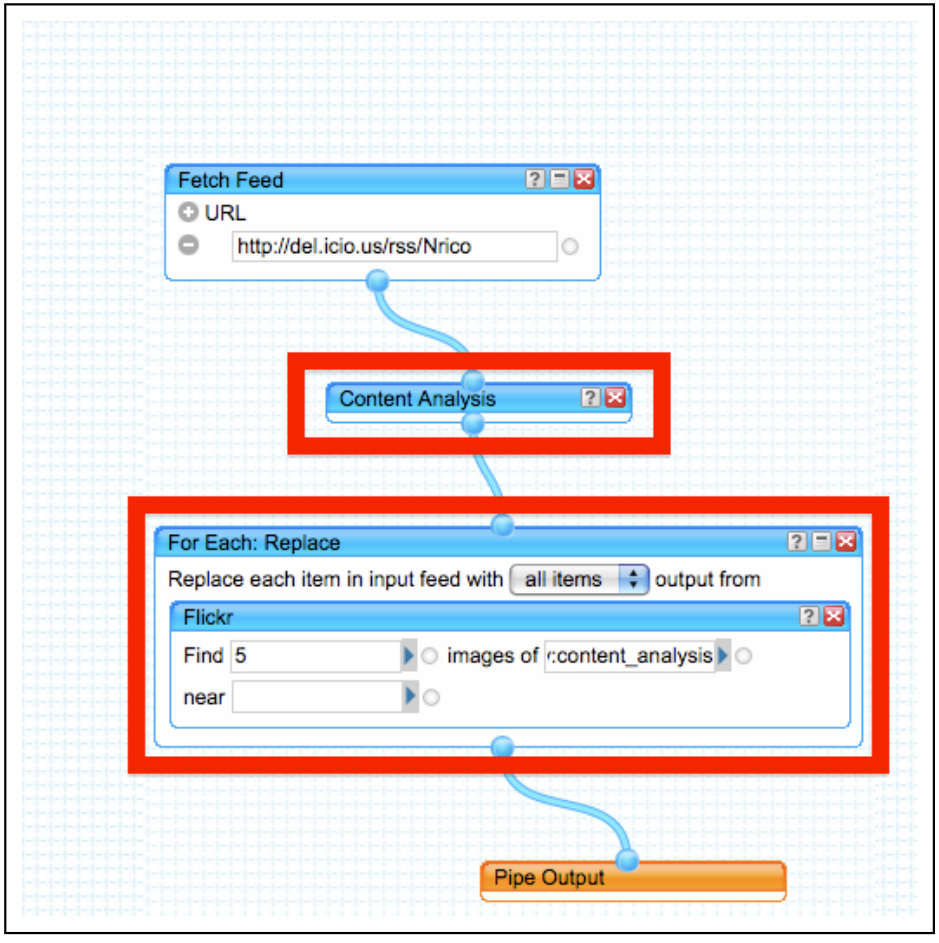
A module that is no longer supported by the environment

Why is this Smelly?

Module may cause unexpected or faulty behavior

Insight

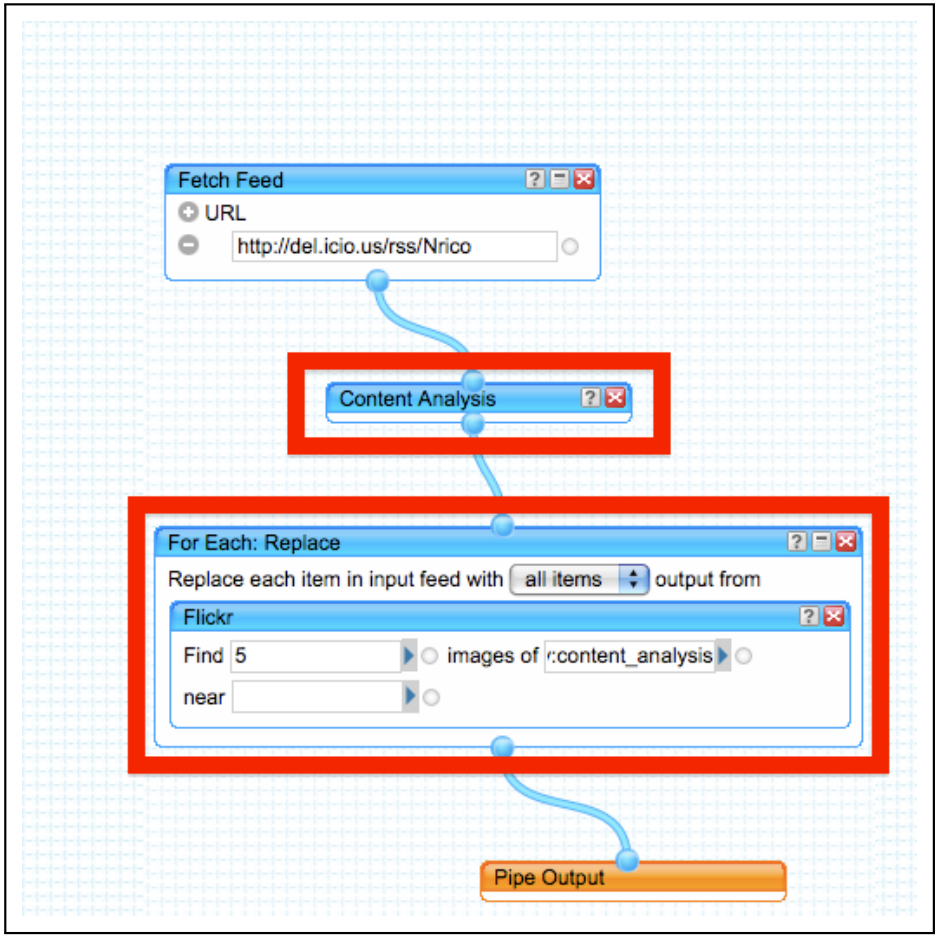
Language resources are independently created and managed



Environmental and Deprecation

Refactor: Replace Deprecated Module

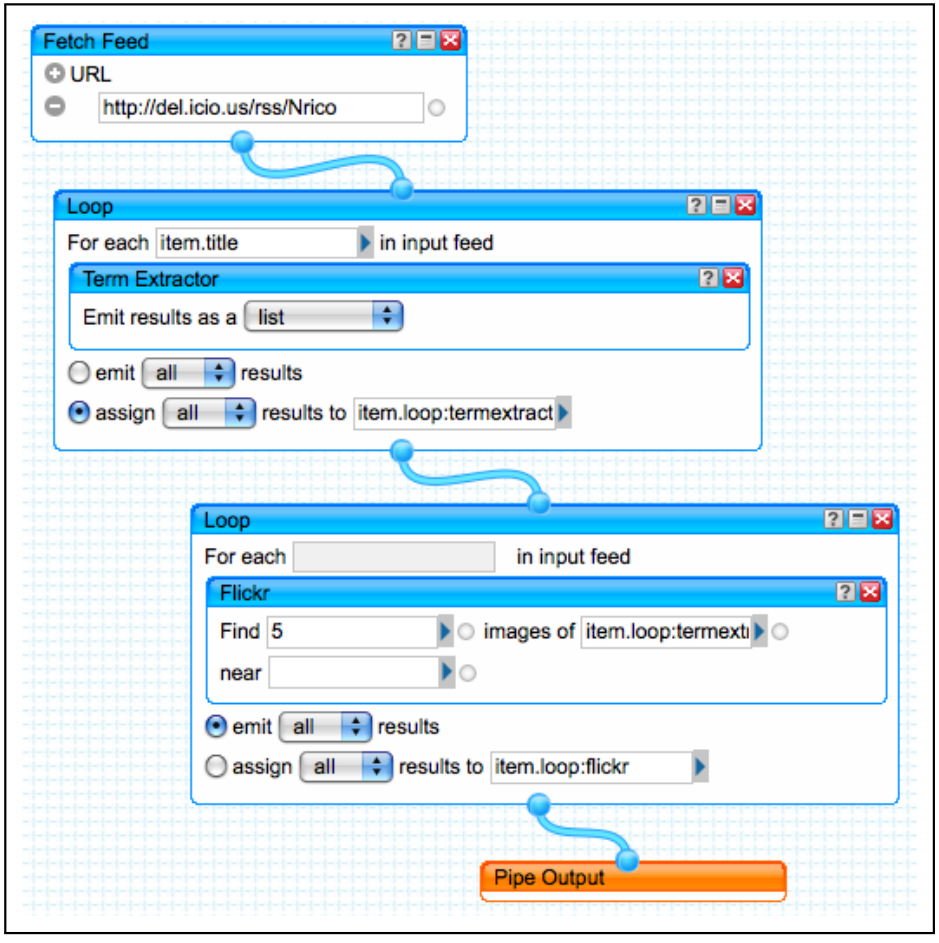
Replace all deprecated modules with their semantic equivalents



Environmental and Deprecation

Refactor: Replace Deprecated Module

Replace all deprecated modules with their semantic equivalents



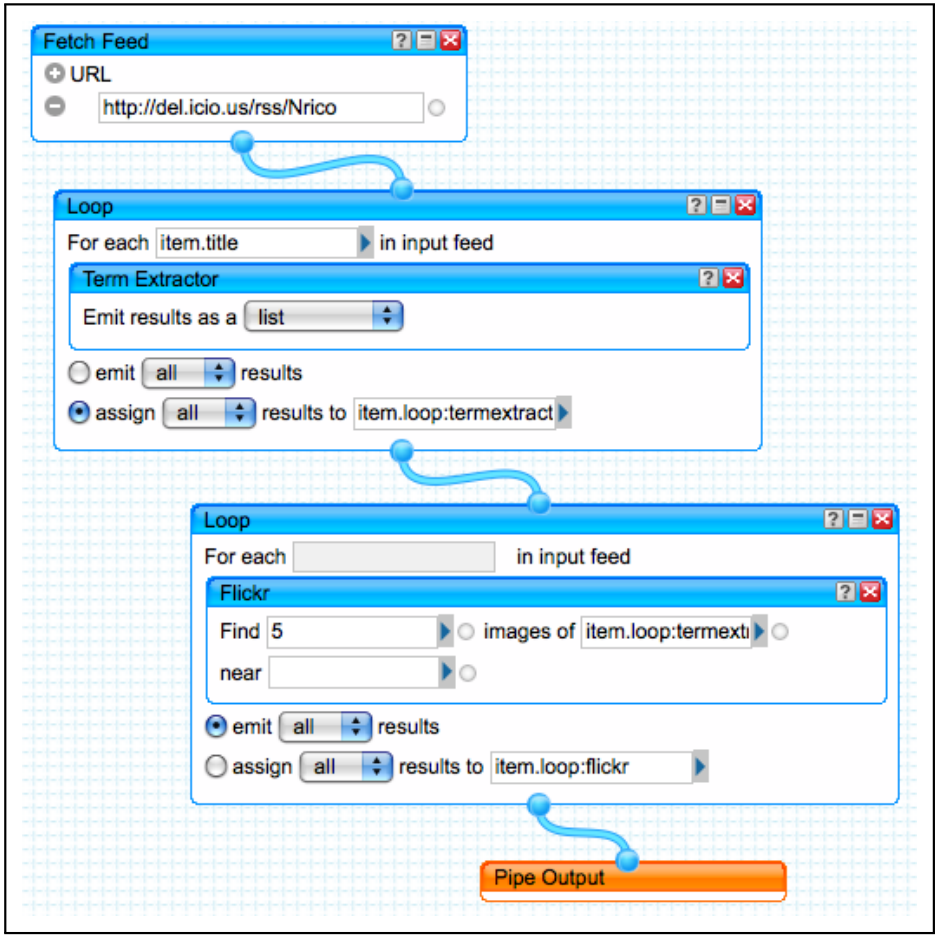
Environmental and Deprecation

Refactor: Replace Deprecated Module

Replace all deprecated modules with their semantic equivalents

Inspiration

Previous work that uses refactoring to update references to deprecated Java libraries*



I. Balaban, F. Tip, and R. Fuhrer. Refactoring support for class library migration. In OOPSLA, 2005.

Smell Types

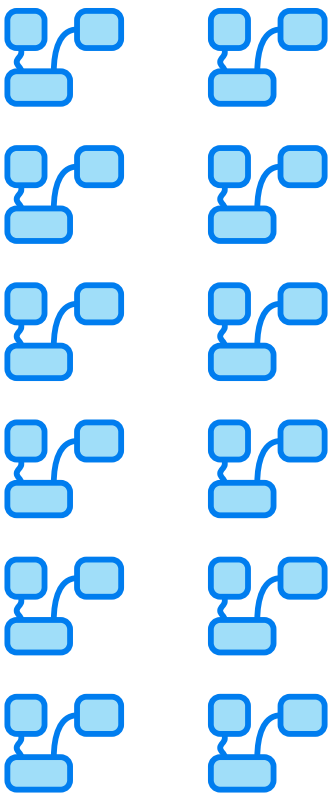
- Laziness – ineffectual or unnecessary components
- Redundancy – duplicate components
- Environmental – external changes cause impact
- **Population-Based – deviants from community norms**

Leveraging a Community of Resources:

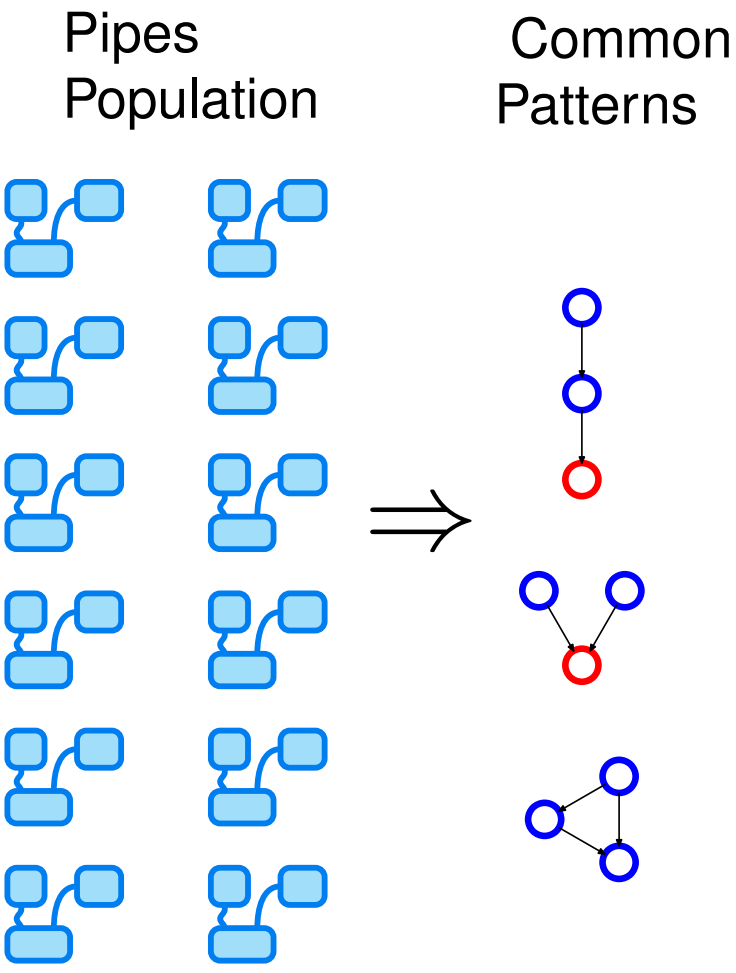
- Popular resources can represent accepted “standards”
- Common patterns may indicate a need for additional community resources

Population-Based Refactorings

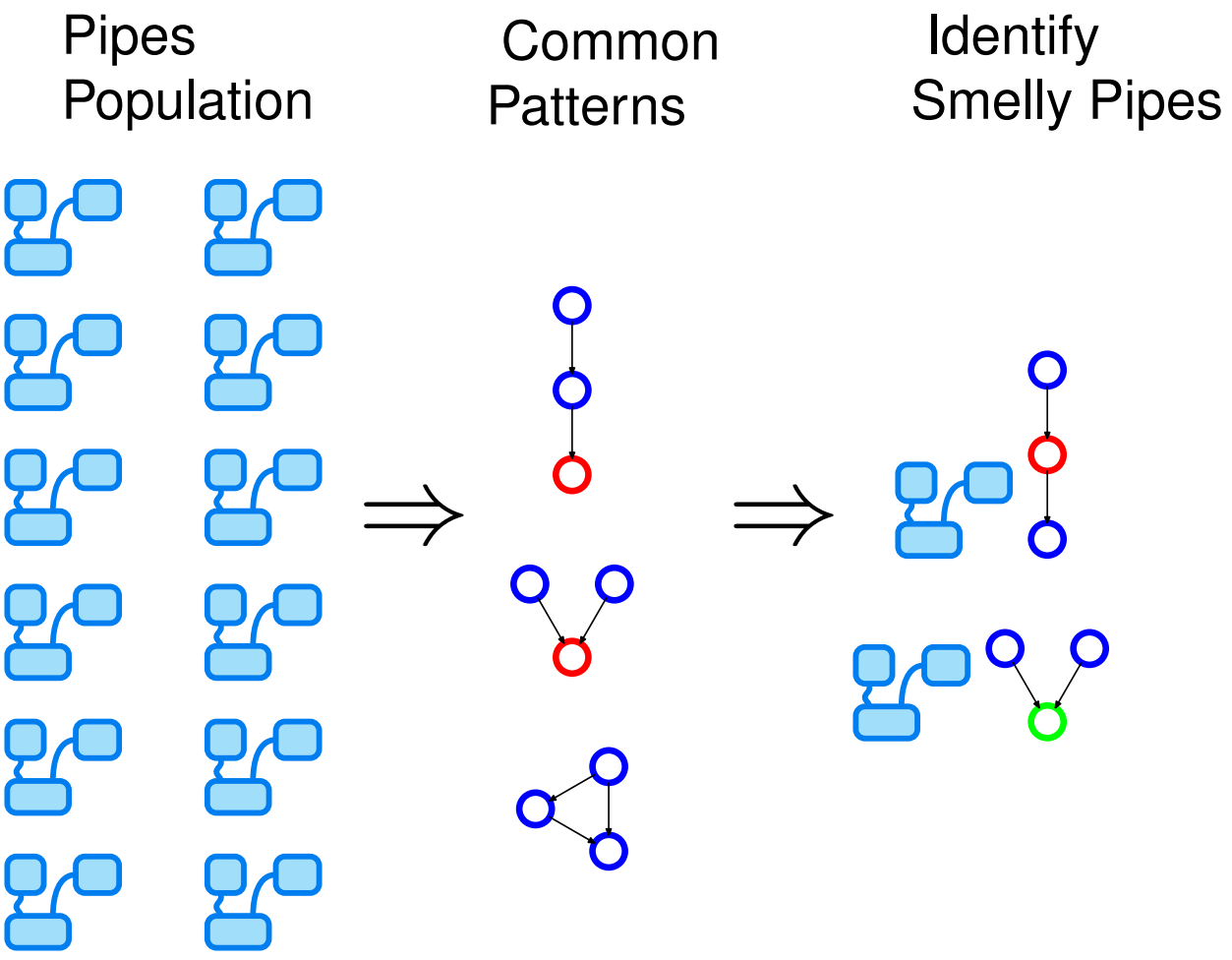
Pipes
Population



Population-Based Refactorings



Population-Based Refactorings



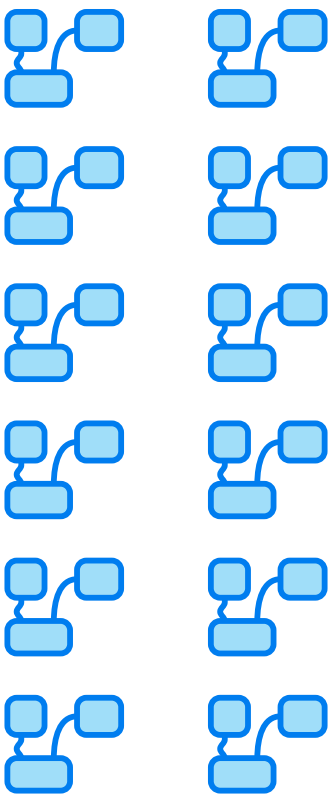
Population-Based Refactorings

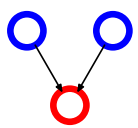
Pipes Population

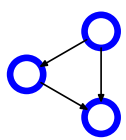
Common Patterns

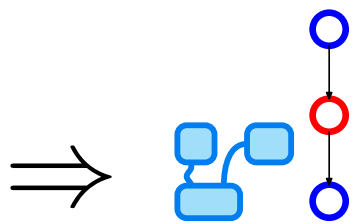
Identify Smelly Pipes

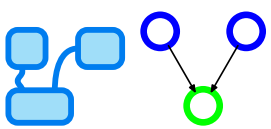
Refactor to Standardize









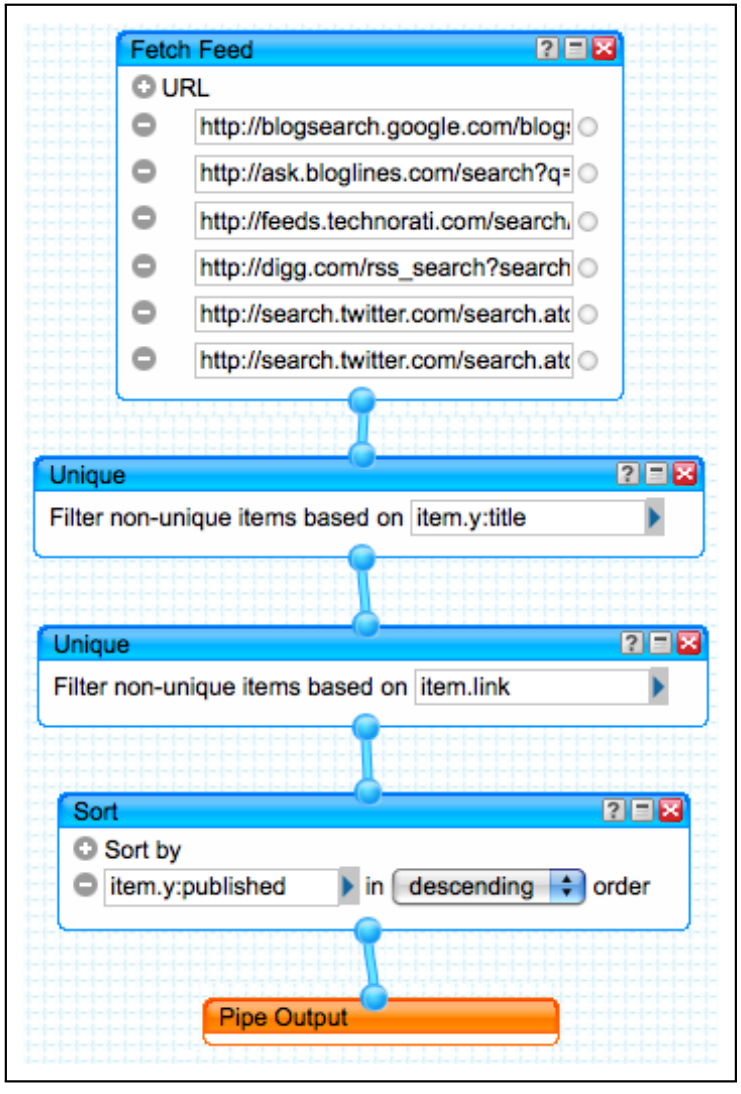


- Normalize Order of Operators
- Extract Global Subpipe
- . . .

Population-Based

Smell: Global Isomorphic Path

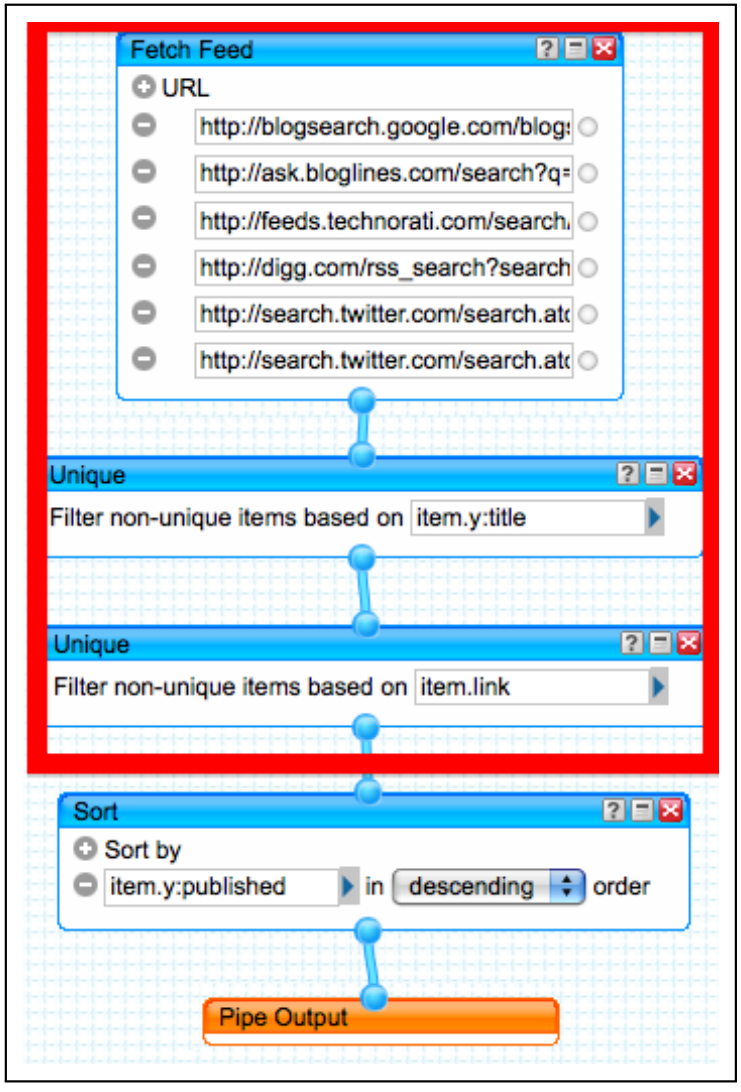
Paths that are isomorphic among pipes in the population



Population-Based

Smell: Global Isomorphic Path

Paths that are isomorphic among pipes in the population



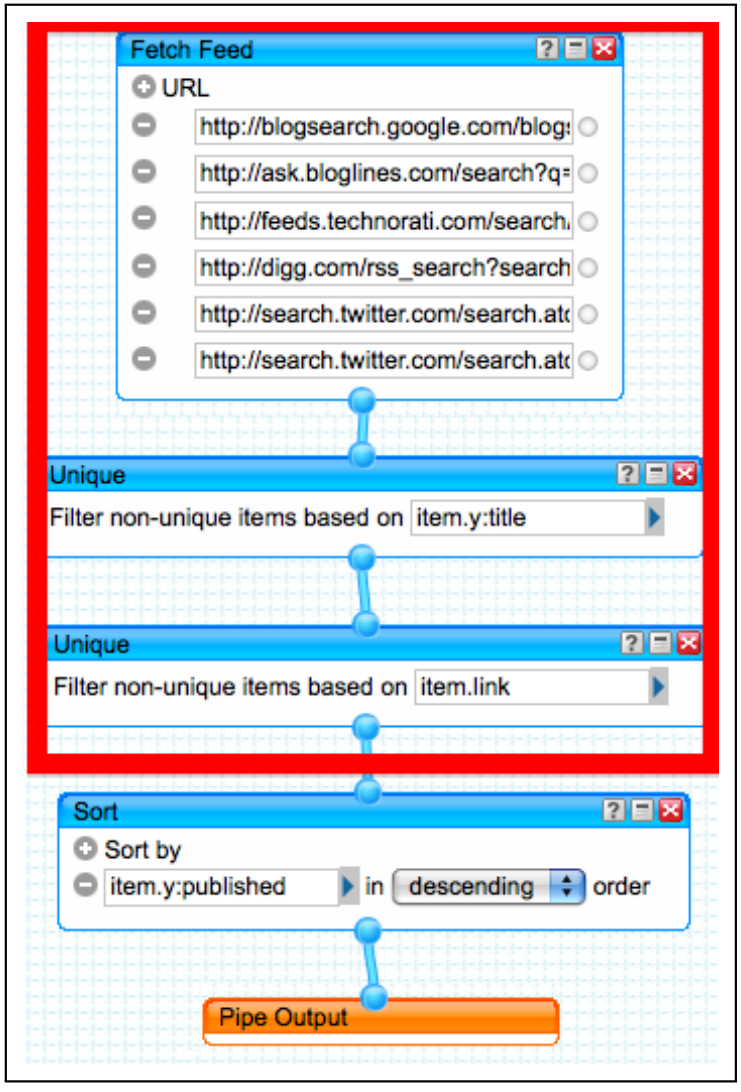
Population-Based

Smell: Global Isomorphic Path

Paths that are isomorphic among pipes in the population

Why is this Smelly?

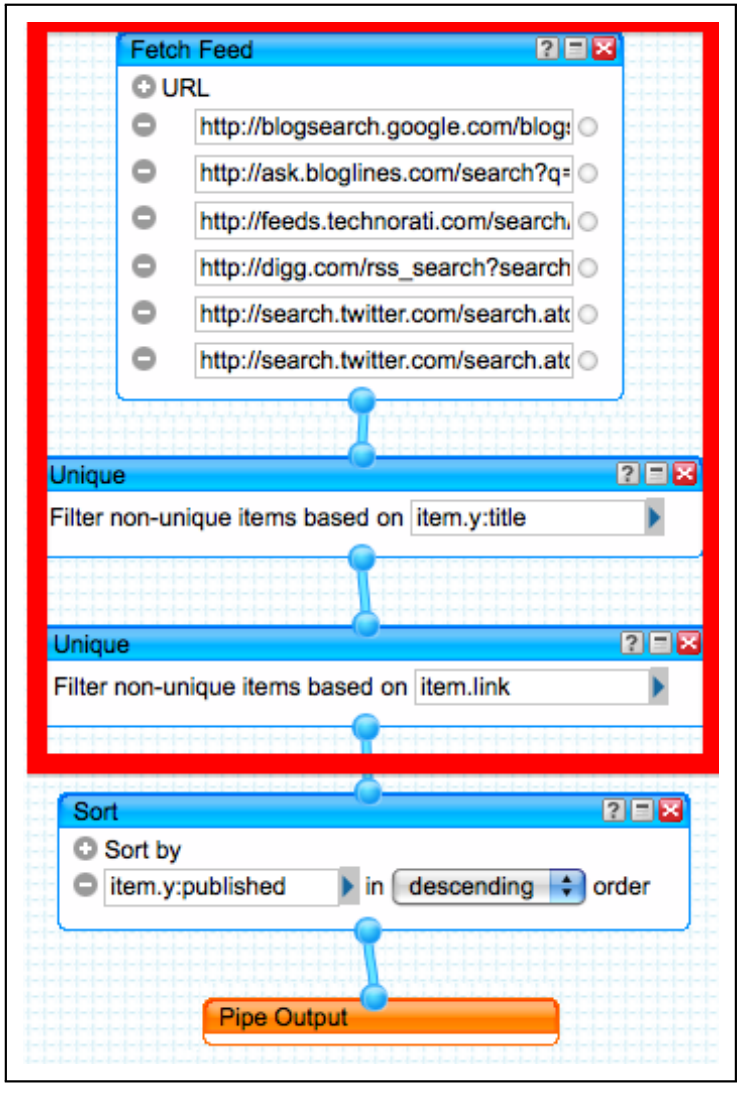
Missed opportunity for abstraction; may inhibit reuse



Population-Based

Refactor: Extract Global Subpipe

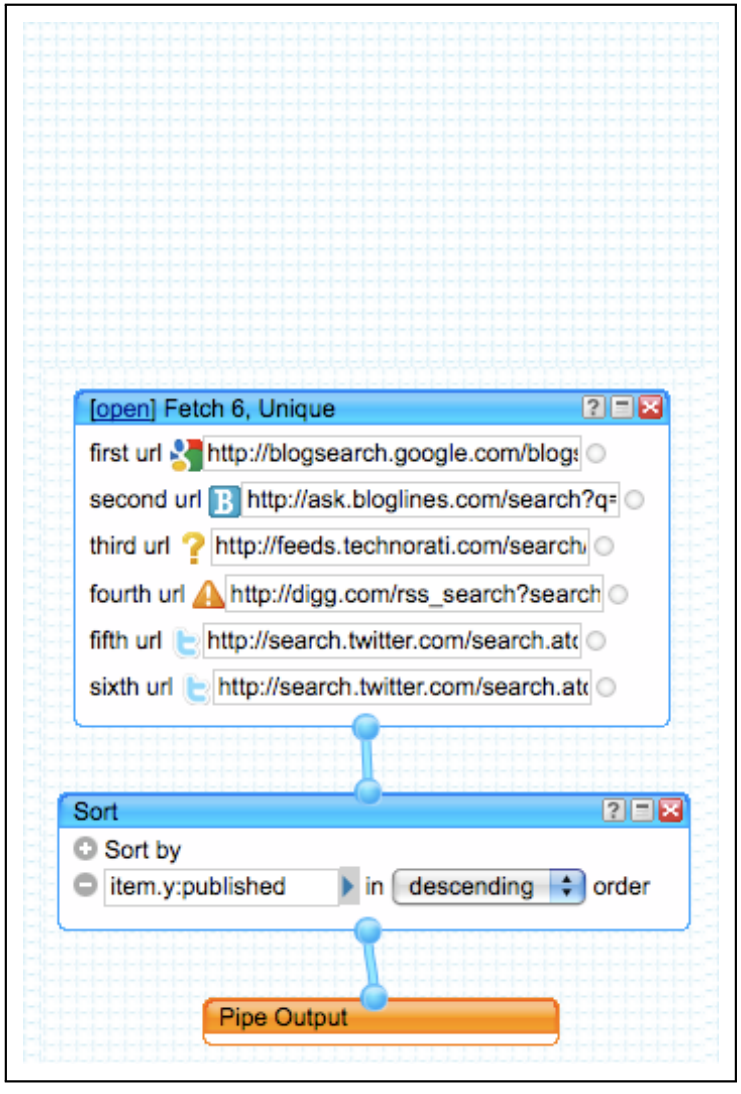
Replaces the isomorphic path with a subpipe from the population



Population-Based

Refactor: Extract Global Subpipe

Replaces the isomorphic path with a subpipe from the population



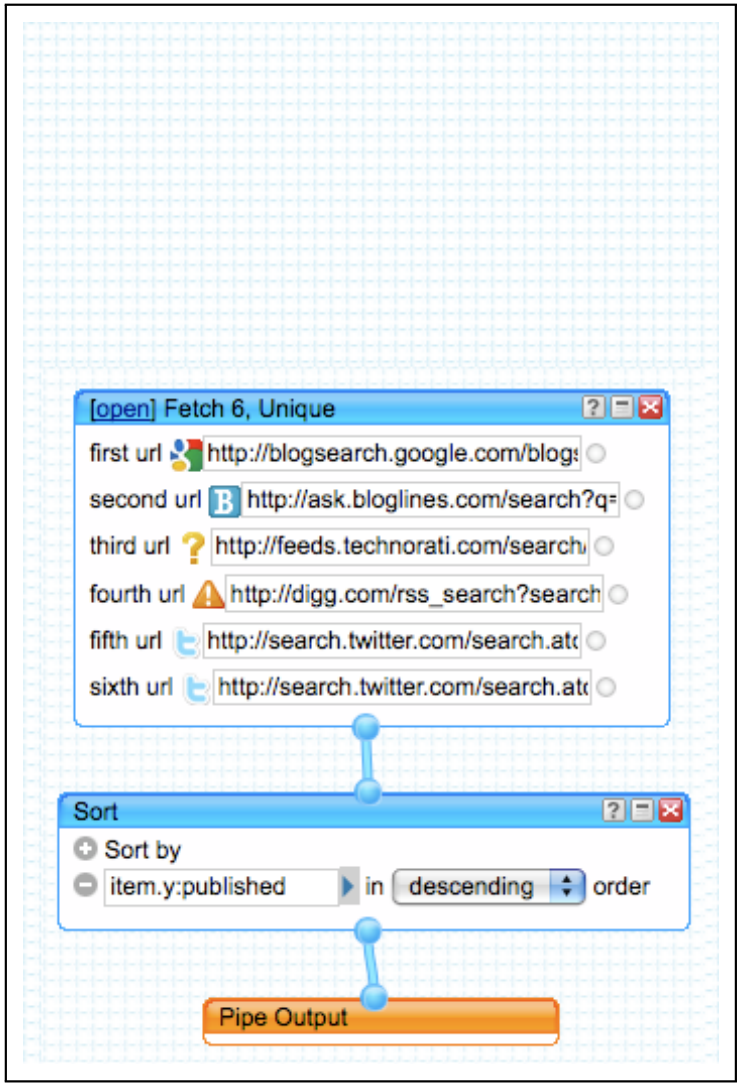
Population-Based

Refactor: Extract Global Subpipe

Replaces the isomorphic path with a subpipe from the population

Inspiration

There is much overlap and clutter across artifacts in end user repositories



Taking Stock: Smells

Smell Type	**Smell**	**Inspiration**
Laziness	Noisy Module Unnecessary Module Unnecessary Abstraction	*Lazy Class*
Redundancy	Duplicate Strings Duplicate Modules Isomorphic Paths	*Duplicate Code* —
Environmental	Deprecated Module Invalid Source	*Java API* —
Population-Based	Non-Conforming Ordering Global Isomorphic Paths	*Coding standards* —

Taking Stock: Smells

Smell Type	Smell	Inspiration
Laziness	Noisy Module **Unnecessary Module** Unnecessary Abstraction	*Lazy Class*
Redundancy	Duplicate Strings **Duplicate Modules** Isomorphic Paths	*Duplicate Code* —
Environmental	Deprecated Module **Invalid Source**	*Java API* —
Population-Based	Non-Conforming Ordering **Global Isomorphic Paths**	*Coding standards* —

Taking Stock: Refactoring

Refactoring Type	Refactoring	Inspiration
Reductions	Clean Up Module	*Remove Parameter*
	Remove Non-Contributing	*Lazy Class*
	Push Down Abstraction	*Inline Method*
Consolidations	Merge Redundant Modules	*Inline Class*
	Collapse Duplicate Paths	—
Abstractions	Pull Up Module	*Pull Up Method*
	Extract Local Subpipe	
Deprecations	Replace Deprecated Mod.	*Java API*
	Remove Invalid Sources	—
Population-Based	Normalize Operations	—
	Extract Global Subpipe	—

Taking Stock: Refactoring

Refactoring Type	Refactoring	Inspiration
Reductions	Clean Up Module	*Remove Parameter*
	Remove Non-Contributing	*Lazy Class*
	Push Down Abstraction	*Inline Method*
Consolidations	Merge Redundant Modules	*Inline Class*
	Collapse Duplicate Paths	—
Abstractions	Pull Up Module	*Pull Up Method*
	Extract Local Subpipe	
Deprecations	Replace Deprecated Mod.	*Java API*
	Remove Invalid Sources	—
Population-Based	Normalize Operations	—
	Extract Global Subpipe	—

Refactoring 8,051 Pipes Programs

		Max % Removed	
	Frequency	**Individually**	**Collectively**
Noisy Module	28%	18%	43%
Unnecessary Module	13%	100%	100%
Unnecessary Abstraction	12%	100%	100%
Duplicate Strings	32%	100%	100%
Duplicate Module	23%	72%	90%
Isomorphic Paths	7%	100%	100%
Deprecated Module	18%	100%	100%
Invalid Source	14%	99%	99%
Non-Conforming Op Order	19%	100%	100%
Global Isomorphic Paths	6%	100%	100%
All Smells	**81%**	**–**	**80%**

Do Smells Matter?

Empirical Study

- 14 'end-user' participants
- Crowdsourced using Mechanical Turk
- All participants demonstrated basic competence in Yahoo! Pipes
- Assumed random assignment of tasks

Empirical Study

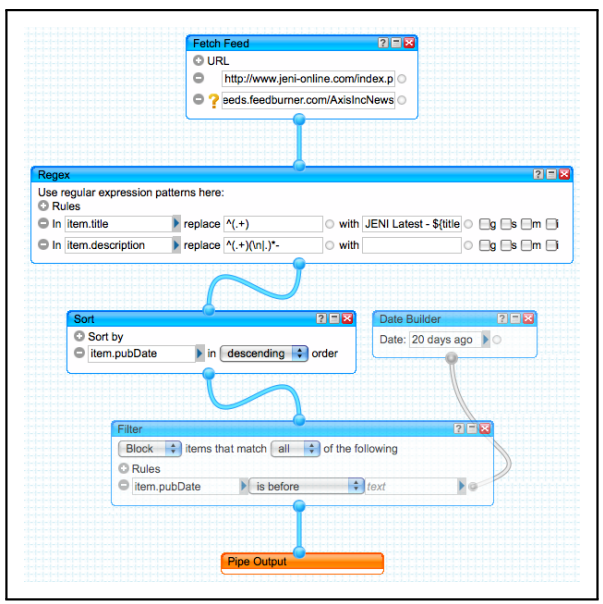
Research Question I

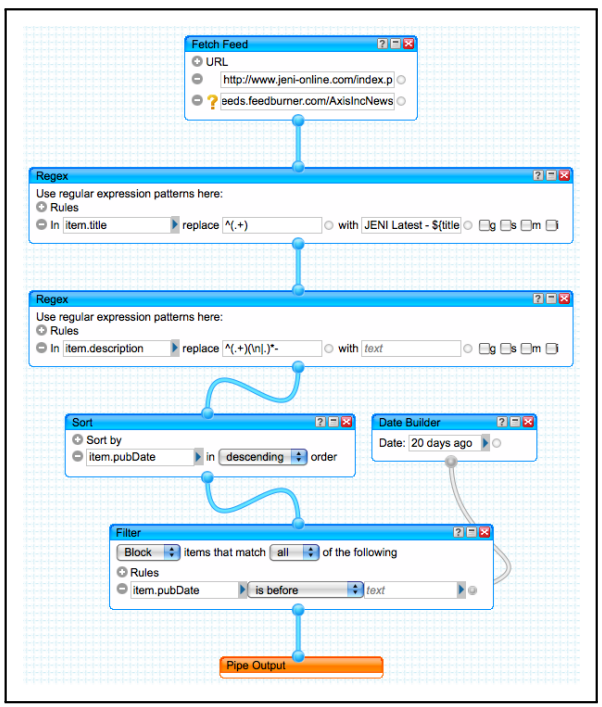
Are pipes with smells **less preferable** than pipes without smells?

Empirical Study

Research Question I

Are pipes with smells **less preferable** than pipes without smells?



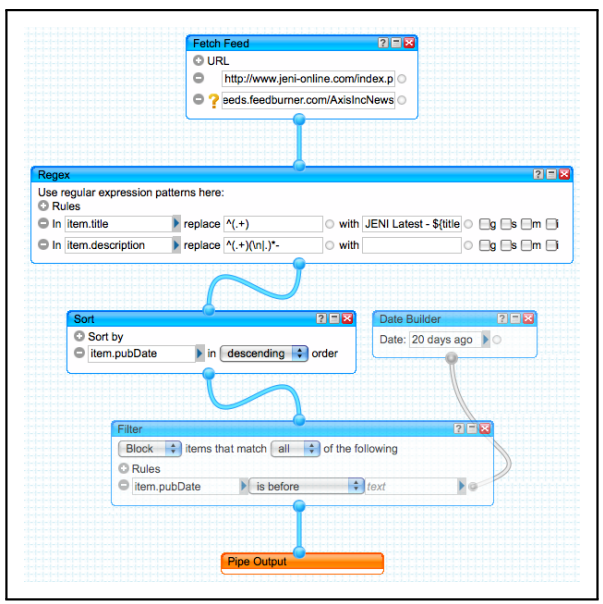


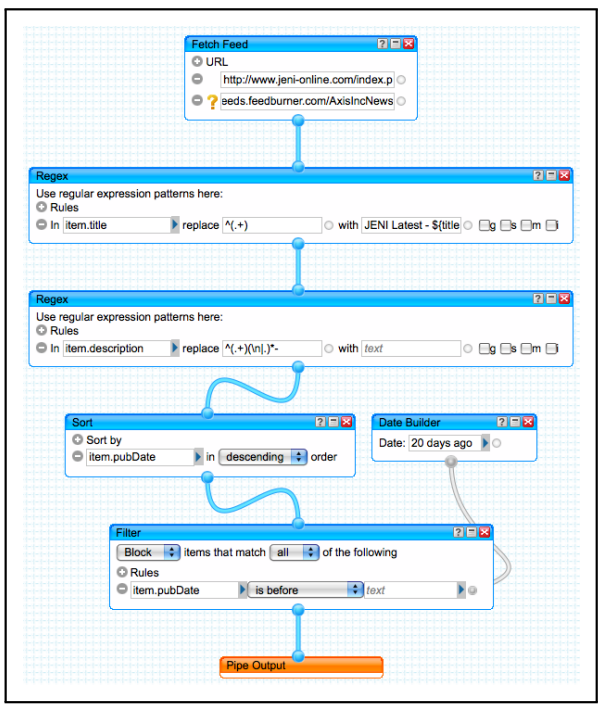
Question: Select the pipe that is easiest *to understand*

Empirical Study

Research Question I

Are pipes with smells **less preferable** than pipes without smells?





Is the refactored pipe preferred?

Yes:	**63%**
No:	24%
Unsure:	13%

Question: Select the pipe that is easiest *to understand*

RQ I: Results Summary

Smell Type + Refactoring	Question	Non-Smelly
Redundancy + Consolidation	Understanding	90%
Redundancy + Abstraction	Maintainability	90%
Redundancy + Consolidation	Understanding	50%
Laziness + Reduction	Maintainability	73%

RQ I: Results Summary

Smell Type + Refactoring	Question	Non-Smelly
Redundancy + Consolidation	Understanding	90%
Redundancy + Abstraction	Maintainability	90%
Redundancy + Consolidation	Understanding	50%
Laziness + Reduction	Maintainability	73%
Environ. + Deprecated Src.	Maintainability	38%
Environ. + Deprecated Mod.	Understanding	33%

RQ I: Results Summary

Smell Type + Refactoring	Question	Non-Smelly
Redundancy + Consolidation	Understanding	90%
Redundancy + Abstraction	Maintainability	90%
Redundancy + Consolidation	Understanding	50%
Laziness + Reduction	Maintainability	73%
Environ. + Deprecated Src.	Maintainability	38%
Environ. + Deprecated Mod.	Understanding	33%
Population (Global Path)	Understanding	33%
Population (Operator Ordering)	Others to Understand	88%

Empirical Study

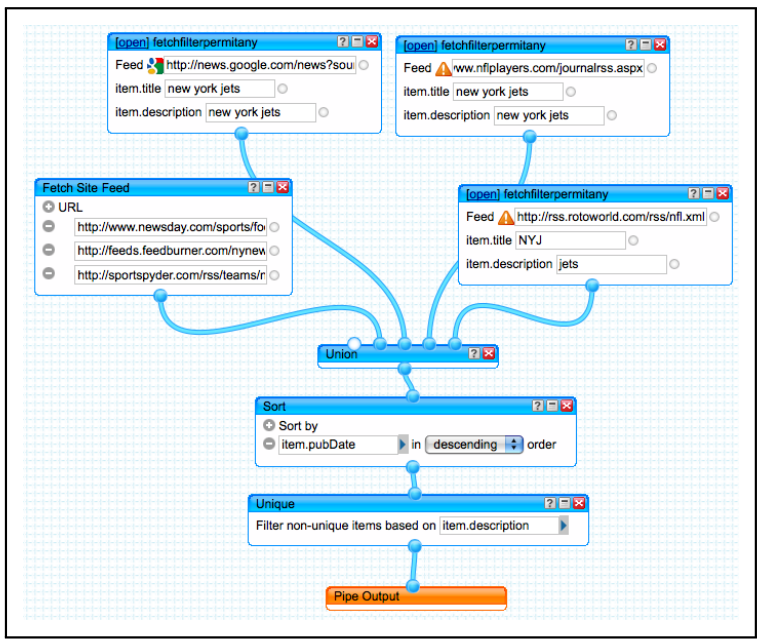
Research Question II

Are smelly pipes **less understandable** than pipes without smells?

Empirical Study

Research Question II

Are smelly pipes **less understandable** than pipes without smells?



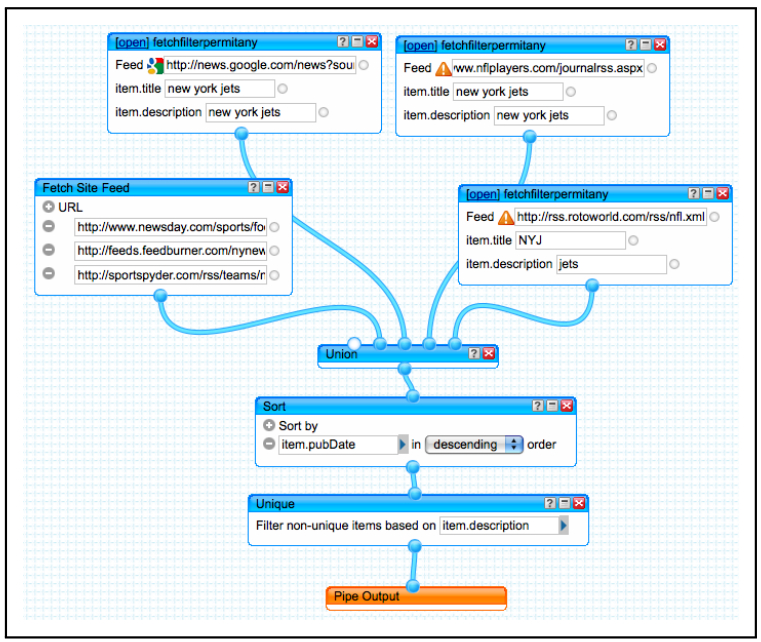
Select the pipe's output:

1. No Output
2. All of the content of the websites specied in the URL Builders.
3. The content of eight websites, filtered based on the presence of a user-defined value in the title of each item.
4. The content of four websites, filtered based on the presence of a user-defined value in the title of each item.

Empirical Study

Research Question II

Are smelly pipes **less understandable** than pipes without smells?



Select the pipe's output:

1. No Output
2. All of the content of the websites specied in the URL Builders.
3. The content of eight websites, filtered based on the presence of a user-defined value in the title of each item.
4. The content of four websites, filtered based on the presence of a user-defined value in the title of each item.

Correctly answered tasks:

Non-Smelly	**80%**
Smelly	67%

Motivations to Refactor

Professional Programmers

- Reduce Complexity
- Increase Understandability
- Create cleaner code
- Improve software design
- Permit more effective testing
- Better maintainability
- Update deprecated library calls
- Parallelization
- Increase code reuse
- . . .

End User Programmers*

* Taken from responses in user study

- Reduce complexity
- Increase Understandability
- Create cleaner code
- Standardize code to familiar patterns from community
- . . .

Motivations to Refactor

Professional Programmers

- Reduce Complexity
- Increase Understandability
- Create cleaner code
- Improve software design
- Permit more effective testing
- Better maintainability
- Update deprecated library calls
- Parallelization
- Increase code reuse
- . . .

End User Programmers*

* Taken from responses in user study

- Reduce complexity
- Increase Understandability
- Create cleaner code
- Standardize code to familiar patterns from community
- . . .

Taking Stock

What we learned:

- Pipe-like mashups are littered with smells
- Refactorings are effective at removing smells
- End users prefer pipes without smells

Taking Stock

What we learned:

- Pipe-like mashups are littered with smells
- Refactorings are effective at removing smells
- End users prefer pipes without smells

What we don't know:

- Will end-user programmers adopt a refactoring tool?
- How can refactoring be extended to other domains?
- Can other end-user domains create opportunities for new refactorings?

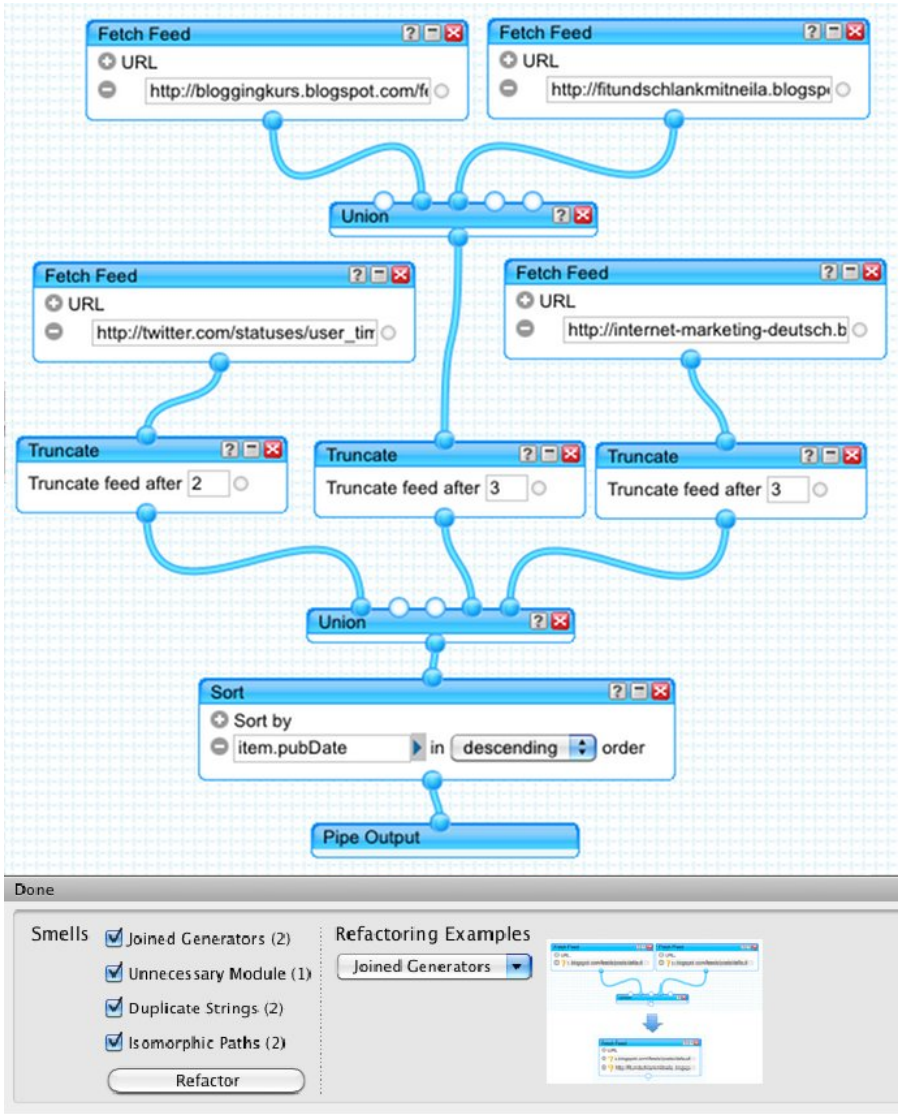
In this talk

- End-user software engineering
- Refactoring for web mashups
- Mashup languages create opportunities to extend refactorings
- Speculate how refactoring adapts to other end user domains

End User Refactoring Engine for Yahoo! Pipes

Idea:

- Integrated into the browser (no extra tool)
- Shows preview of the result for each refactoring (no surprises)
- Offers to perform refactorings for the end user (no extra effort)



Web Macros

Environments: CoScripter, . . .

Programs: log a user into a system, insert information from spreadsheet to form, . . .

Refactoring in Web Macros

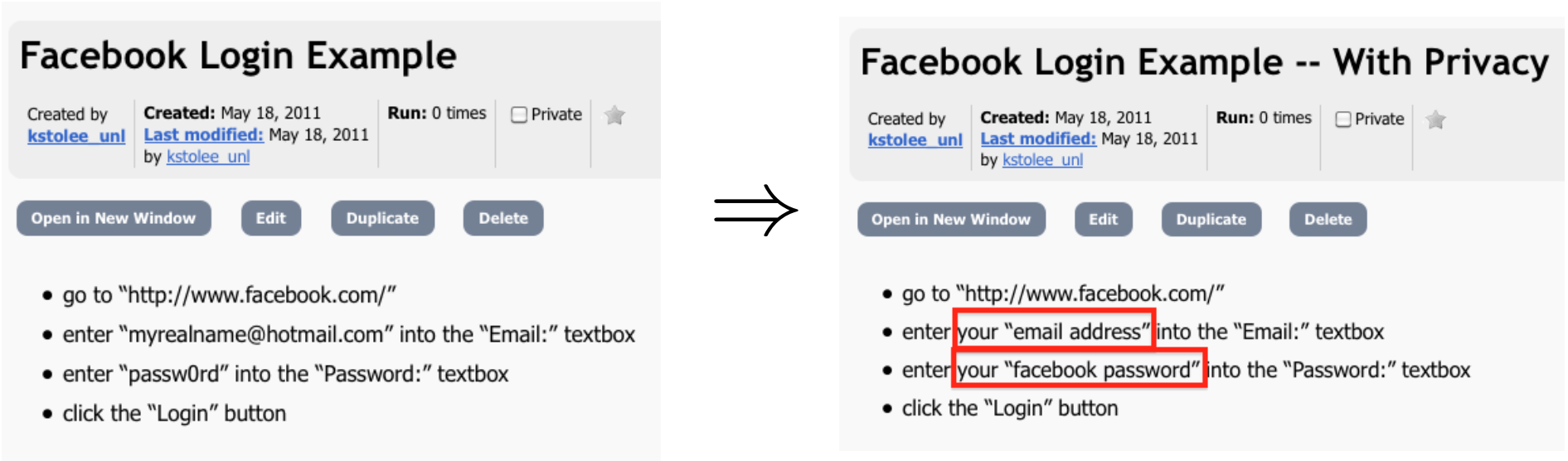
Refactor for Privacy and Flexibility:

1. User records a macro with username and password
2. User wants to share the macro, but maintain privacy
3. Refactor to reference username and password in personal database

Refactoring in Web Macros

Refactor for Privacy and Flexibility:

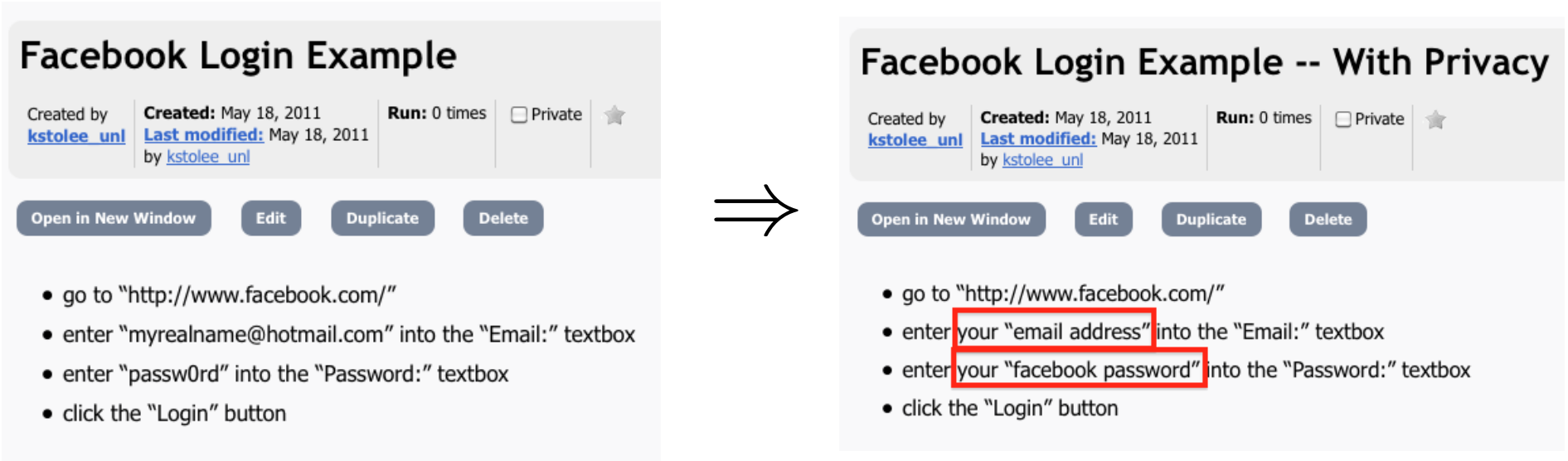
1. User records a macro with username and password
2. User wants to share the macro, but maintain privacy
3. Refactor to reference username and password in personal database



Refactoring in Web Macros

Refactor for Privacy and Flexibility:

1. User records a macro with username and password
2. User wants to share the macro, but maintain privacy
3. Refactor to reference username and password in personal database

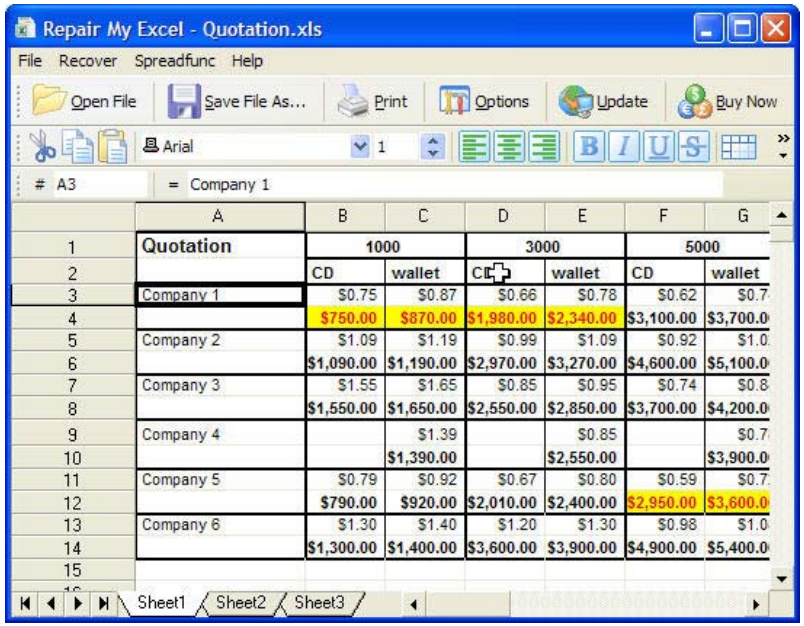


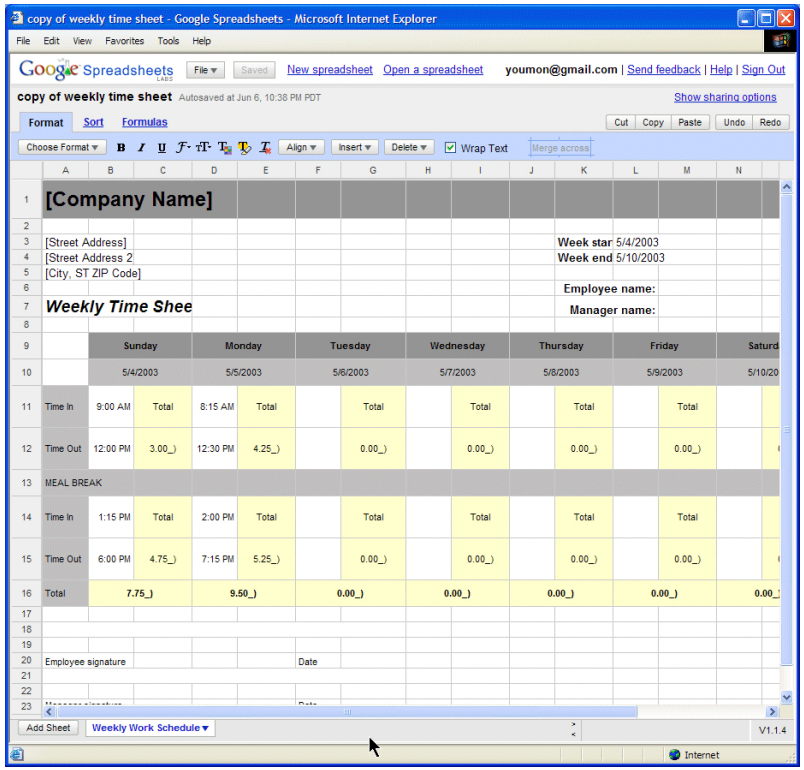
Other opportunities: performance, parallelization, invalid sources, cleanliness

Spreadsheets

Environments: Excel, Google Spreadsheet, . . .

Programs: Accounting worksheet, event planning, financial forecasting, . . .





Refactoring in Excel

Refactor for Standardization to Community:

1. User creates accounting spreadsheet like this:

A	B	C	D	E	F
Reference	**Date**	**Description**	**Balance**	**Credit**	**Debit**
Target	5/17/2011	Clothing, Shoes	$1,279.89	$ 65.43	
Shell	5/17/2011	gas	$1,236.34	$ 43.55	
Express	5/17/2011	refund for shirt	$1,269.54		$ 33.20

Refactoring in Excel

Refactor for Standardization to Community:

1. User creates accounting spreadsheet like this:

A	B	C	D	E	F
Reference	**Date**	**Description**	**Balance**	**Credit**	**Debit**
Target	5/17/2011	Clothing, Shoes	$1,279.89	$ 65.43	
Shell	5/17/2011	gas	$1,236.34	$ 43.55	
Express	5/17/2011	refund for shirt	$1,269.54		$ 33.20

2. Analysis of community spreadsheets shows most others like:

Date	Description	Reference	Debit	Credit	Balance
5/17/2011	Clothing, Shoes	Target		$ 65.43	$1,279.89

Date	Description	Reference	Debit	Credit	Balance
5/17/2011	refund	Banana Rep		$ 65.43	$1,279.89

Date	Description	Reference	Debit	Credit	Balance
5/17/11	lunch	Buffet		$ 7.50	$1,337.82

Date	Description	Reference	Debit	Credit	Balance
5/17/2011	gas	Shell		$ 65.43	$1,279.89

Refactoring in Excel

Refactor for Standardization to Community:

1. User creates accounting spreadsheet like this:

Reference	Date	Description	Balance	Credit	Debit
Target	5/17/2011	Clothing, Shoes	$1,279.89	$ 65.43	
Shell	5/17/2011	gas	$1,236.34	$ 43.55	
Express	5/17/2011	refund for shirt	$1,269.54		$ 33.20

2. Analysis of community spreadsheets shows most others like:

Date	Description	Reference	Debit	Credit	Balance
5/17/2011	Clothing, Shoes	Target		$ 65.43	$1,279.89

Date	Description	Reference	Debit	Credit	Balance
5/17/2011	refund	Banana Rep		$ 65.43	$1,279.89

Date	Description	Reference	Debit	Credit	Balance
5/17/11	lunch	Buffet		$ 7.50	$1,337.82

Date	Description	Reference	Debit	Credit	Balance
5/17/2011	gas	Shell		$ 65.43	$1,279.89

3. Reorder columns to fit community

Date	Description	Reference	Debit	Credit	Balance
5/17/2011	Clothing, Shoes	Target		$ 65.43	$1,279.89
5/17/2011	gas	Shell		$ 43.55	$1,236.34
5/17/2011	refund for shirt	Express	$ 33.20		$1,269.54

Refactoring in Excel

Refactor for Standardization to Community:

1. User creates accounting spreadsheet like this:

A	B	C	D	E	F
Reference	**Date**	**Description**	**Balance**	**Credit**	**Debit**
Target	5/17/2011	Clothing, Shoes	$1,279.89	$ 65.43	
Shell	5/17/2011	gas	$1,236.34	$ 43.55	
Express	5/17/2011	refund for shirt	$1,269.54		$ 33.20

2. Analysis of community spreadsheets shows most others like:

Date	Description	Reference	Debit	Credit	Balance
5/17/2011	Clothing, Shoes	Target		$ 65.43	$1,279.89

Date	Description	Reference	Debit	Credit	Balance
5/17/2011	refund	Banana Rep		$ 65.43	$1,279.89

Date	Description	Reference	Debit	Credit	Balance
5/17/11	lunch	Buffet		$ 7.50	$1,337.82

Date	Description	Reference	Debit	Credit	Balance
5/17/2011	gas	Shell		$ 65.43	$1,279.89

3. Reorder columns to fit community

A	B	C	D	E	F
Date	**Description**	**Reference**	**Debit**	**Credit**	**Balance**
5/17/2011	Clothing, Shoes	Target		$ 65.43	$1,279.89
5/17/2011	gas	Shell		$ 43.55	$1,236.34
5/17/2011	refund for shirt	Express	$ 33.20		$1,269.54

Other opportunities: maintenance, text to drop-down, remove temporary variables

Educational Languages

Environments: Scratch, Alice, Kodu, . . .
Programs: Animations, simulations, interactive games, . . .

Refactoring in Kodu

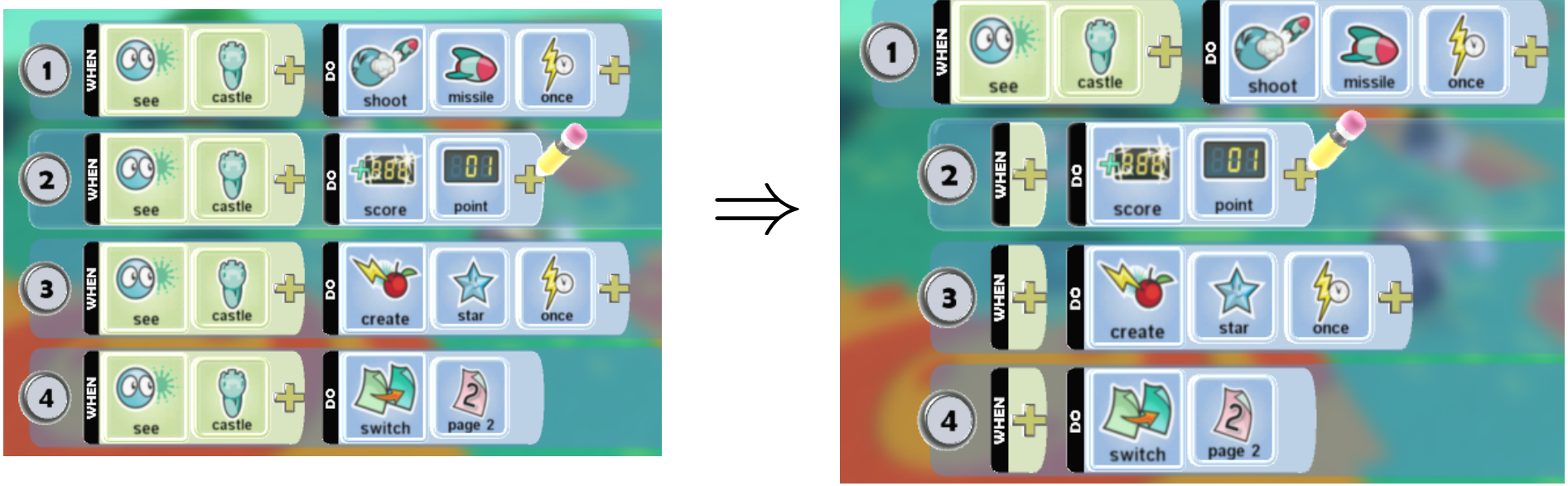
Refactor for Simplicity:

1. User creates many rules with replicated conditions
2. Multiple actions can be bound to a single condition
3. Concatenate multiple actions with a single condition

Refactoring in Kodu

Refactor for Simplicity:

1. User creates many rules with replicated conditions
2. Multiple actions can be bound to a single condition
3. Concatenate multiple actions with a single condition



Refactoring in Kodu

Refactor for Standardization to Community:

1. User creates many rules to define character behavior
2. Most other users prioritize the rules by type (e.g., movement, scoring, special features, . . .)
3. Reorder rules to match community

Refactoring in Kodu

Refactor for Standardization to Community:

1. User creates many rules to define character behavior
2. Most other users prioritize the rules by type (e.g., movement, scoring, special features, . . .)
3. Reorder rules to match community

Refactoring in Kodu

Refactor for Standardization to Community:

1. User creates many rules to define character behavior
2. Most other users prioritize the rules by type (e.g., movement, scoring, special features, . . .)
3. Reorder rules to match community

Other opportunities: remove dead code

In this talk

- End-user software engineering
- Refactoring for web mashups
- Mashup languages create opportunities to extend refactorings
- Speculate how refactoring adapts to other end user domains

In this talk

- End-user software engineering
- Refactoring for web mashups
- Mashup languages create opportunities to extend refactorings
- Speculate how refactoring adapts to other end user domains

This work was supported in part by NSF Graduate Research Fellowship CFDA#47.076, NSF Award #0915526, and AFOSR Award #9550-10-1-0406.