STU FIIT

Software Developer Activity as a Source for Identifying Hidden Source Code Dependencies

Martin Konôpka, Mária Bieliková

Slovak University of Technology in Bratislava

Source code dependencies

```
private static void StoreStreamItems(ISvcCaller svcCaller, LinkedList<Data.LinkedStrea
{
    /// Store the last update date and commit changes to database
    lastUpdate.UpdatedAt = latestNewUpdateDate;
    container.SaveChanges();

    logger.Info(LogHelper.Prepare(string.Format("last update {0} of {1}", latestNewUpd

    if (streamItems.Any() && EndpointClient.TryRegisterStream(StreamManager.ITGStreamU
    {
        EndpointClient.StreamRDFs(StreamManager.ITGStreamUri, streamItems);
    }

    container.Dispose();

    container = new Data.ITGeneratorDataContainer();
    lastUpdate = GetLastUpdate(svcCaller, container);
}
```

Source code dependencies

```
private static void StoreStreamItems(ISvcCaller svcCaller, LinkedList<Data.LinkedStrea
{
    /// Store the last update date and commit changes to database
    lastUpdate.UpdatedAt = latestNewUpdateDate;
    container.SaveChanges();

    logger.Info(LogHelper.Prepare(string.Format("last update {0} of {1}", latestNewUpd

    if (streamItems.Any() && EndpointClient.TryRegisterStream(StreamManager.ITGStreamU
    {
        EndpointClient.StreamRDFs(StreamManager.ITGStreamUri, streamItems);
    }

    container.Dispose();

    container = new Data.ITGeneratorDataContainer();
    lastUpdate = GetLastUpdate(svcCaller, container);
}
```

Dependency graph

Important for development and maintenance

Explicitly stated in source code

Aren't we missing anything?

It's just a static view on code structure

- Parallel interface implementations

Lack of cross-language support

- Client-server code, C# – XML, HTML, configuration files

Mainly for statically typed languages

- C/C++, Java, C#, etc.

And, where are developer's inspirations and intents?

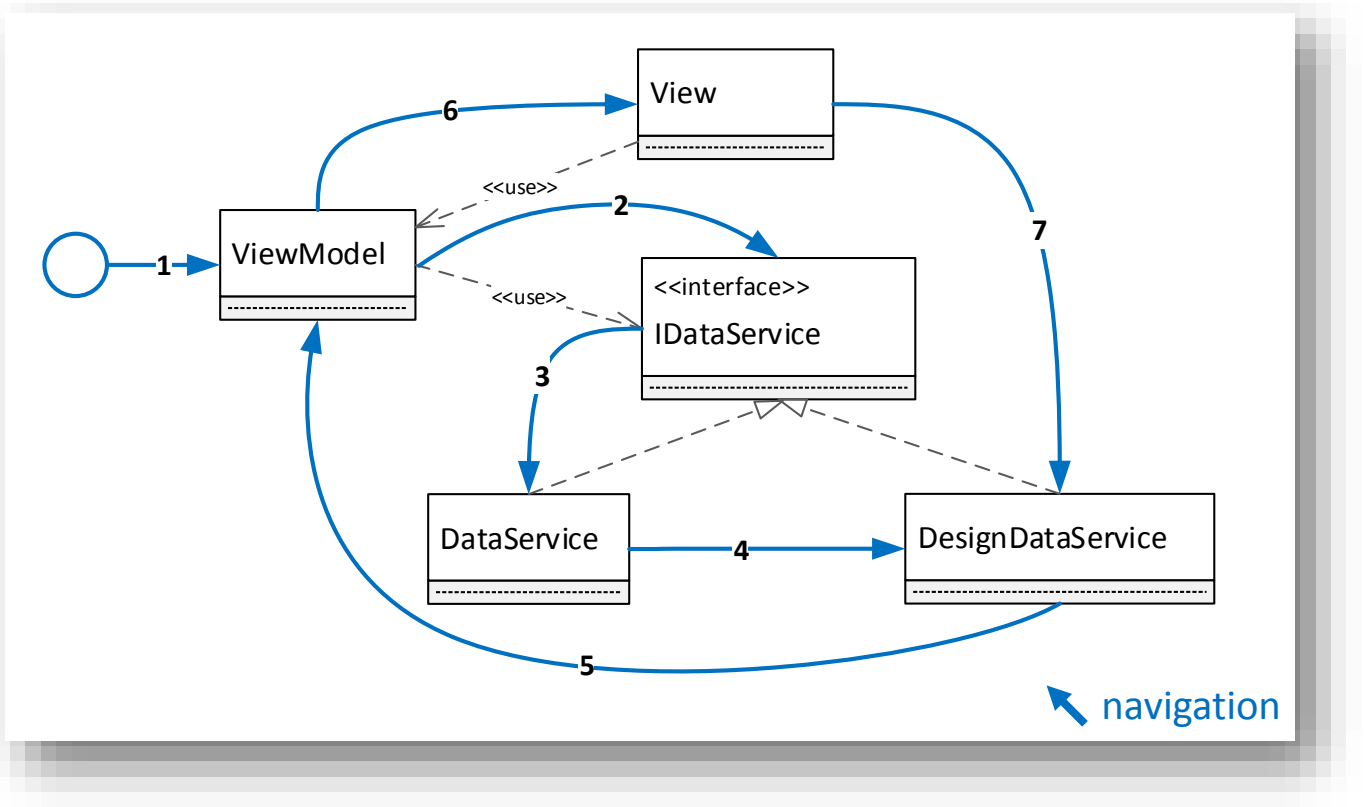
Solution is at hand.

Developer performs *interactions* within *activities* to accomplish a *task*

- Tasks - Add new service XYZ, Fix bug #421
- Activities – studying code, identifying problem, debugging
- Interaction events – open a document, switch-to a document

Monitoring of developers is difficult, but possible

Developer works on a task…



…and possibly reveals (hidden) dependencies

Other activities

Editing of multiple components in parallel
- Even when they are not directly dependent

Copying a code fragment
- To keep consistency when changes are made to one copy
- Code transformations

Configuring components with a config file

Committing a task solution

Inspirations from

Navigation in source code

- To ease program comprehension (DeLine et al., 2005)
- Knowledge of code (Fritz et al., 2014)

Mylyn – task context (Kersten & Murphy, 2006)

- Task-related source code components
- Navigation, code study

Cross-language dependencies

- Dynamic behavior (Spasojević et al., 2014)

Implicit source dependencies

Implicitly arise from developer's activity

Represent *potential* source dependencies

Source code dependencies

Oriented weighted connections

- Components *S*: assembly, package, class, method...

$$D = S \times S \times T \times R \qquad d \in \mathrm{D}$$

$$d = (source, target, timestamp, weight)$$

Set of potential implicit dependencies D_{imp}

- Extending *D* with *property* – information about an interaction

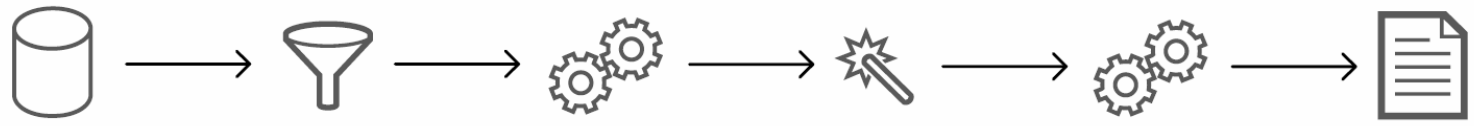
$$d_{imp} = (source, target, timestamp, weight, property)$$

Identification of dependencies D_{imp}

Input: *Interaction events in IDE and RCS*

1. Filtering events for identification
2. Identification of potential dependencies
3. Weighting and validation
4. Dependency graph construction

Output: *Graph of implicit dependencies*



1. Filtering events

Interaction events for potential dependencies

Navigation

- Developer navigates between components very often

Content movement

- Developer moves a code fragment, e.g., copy-paste

Commits

- Committed files possibly relate to the same task

2. Identification

Three kinds of dependencies D_{imp}

- Time-related – navigation in source code

$$d_{imp,time} = (s_1, s_2, t, w, time\ window)$$

- Content-related – content movemenet

$$d_{imp,content} = (s_1, s_2, t, w, content)$$

- Commit-related – committing a set of files

$$d_{imp,commit} = (s_1, s_2, t, w, total\ count)$$

3. Weighting and validation

$$weight\colon D_{imp} \rightarrow \langle 0,1 \rangle$$

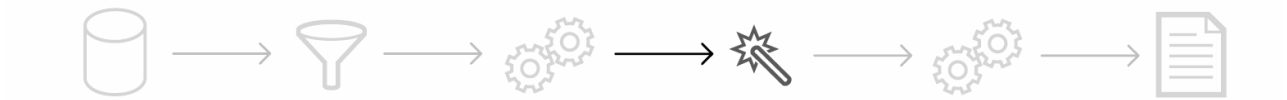
Determining significance for each kind differently

- Property – dwell time, contents and number of committed files

$$validity_{imp}\colon D_{imp} \times T \rightarrow \langle 0,1 \rangle$$

Decay of potential dependencies

For explicit dependencies – yes/no from code



4. Graph construction

Graph of implicit dependencies

$$G = (V, E_{imp})$$

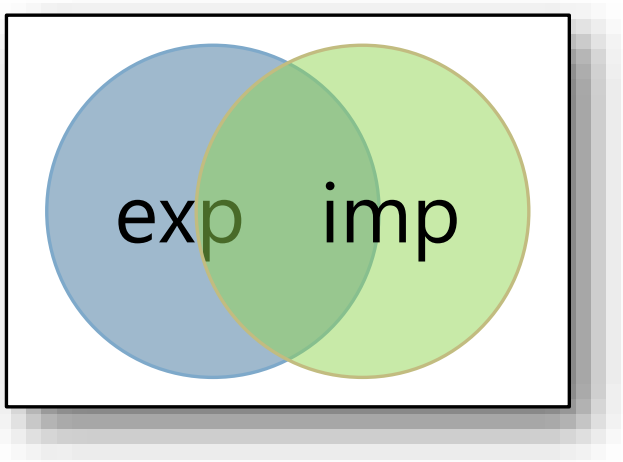
Aggregating D_{imp} into edges E_{imp}

$$e_{imp} = (s_1, s_2, w) \quad e_{imp} \in E_{imp}$$

$$w_{s_1,s_2} = \sum_{d \in D_{imp,s_1,s_2}} validity_{imp}(d, t')\, weight(d)$$

Contribution of D_{imp}

Extending the space of explicit dependencies



Potentially important dependencies

- Independent from programming language
- Task-related connections

Evaluation

Potential implicit vs explicit dependencies

Significance of implicit dependencies for maintenance

Data

- Personalized Conveying of Information and Knowledge
- In cooperation with software company
- http://perconik.fiit.stuba.sk/

Five student projects

- Master courses
- 1 year duration, 2.4k – 5.3k LOC
- Microsoft Visual Studio, C#/.NET & ASP.NET MVC

1. Implicit vs explicit

A. Explicit dependencies among implicit ones

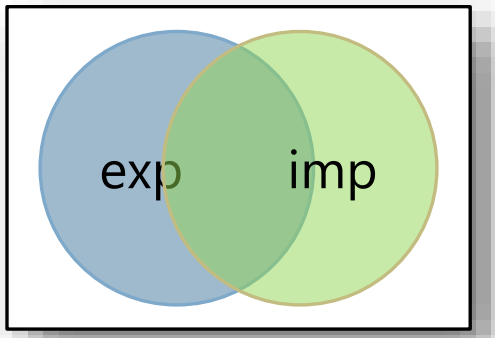
- $w \geq 1$: 32.1% - 53.6%
- $w \geq 2$: 40.1% - 54.0%

$$\frac{|E_{exp} \cap E_{imp}|}{|E_{imp}|}$$

B. Explicit dependencies identified with interactions

- $w \geq 1$: 39.8% - 78.9%
- $w \geq 2$: 22.7% - 57.1%

$$\frac{|E_{exp} \cap E_{imp}|}{|E_{exp}|}$$



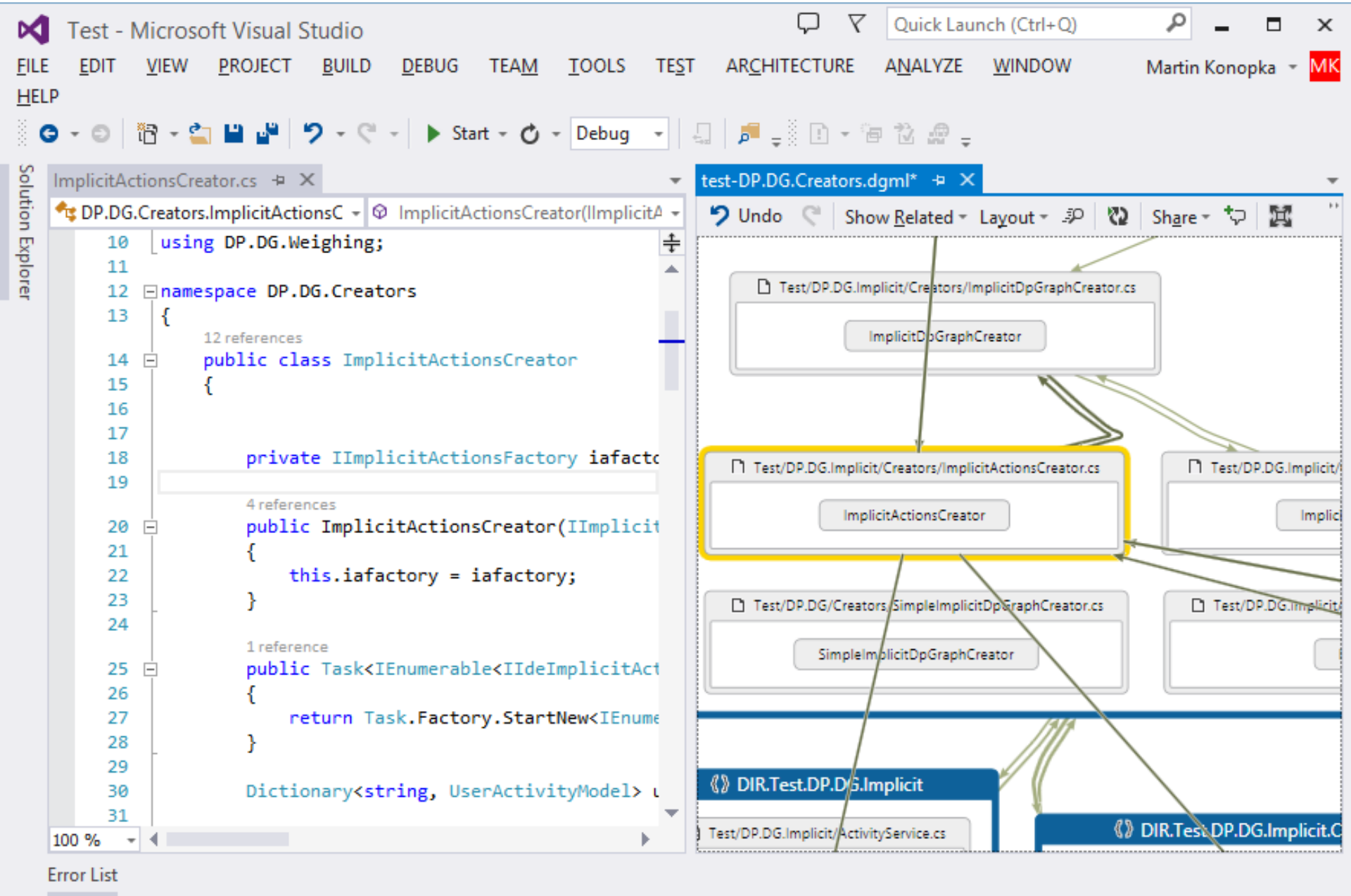
2. Significance of D_{imp}

Are implicit dependencies usable for maintenance?

- Implicit dependency between source code files A and B
- *If components in the source code file A are changed, the contents of the file B should be checked or changed as well.*

Developers manually evaluated each edge

- Viewing DGML files in Visual Studio


Test - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Martin Konopka
Start
Debug
Solution Explorer
ImplicitActionsCreator.cs
DP.DG.Creators.ImplicitActionsC
ImplicitActionsCreator(IImplicitA
using DP.DG.Weighing;
namespace DP.DG.Creators
{
12 references
public class ImplicitActionsCreator
{
private IImplicitActionsFactory iafacto
4 references
public ImplicitActionsCreator(IImplicit
{
this.iafactory = iafactory;
}
1 reference
public Task<IEnumerable<IIdeImplicitAct
{
return Task.Factory.StartNew<IEnume
}
Dictionary<string, UserActivityModel>
100 %
Error List
test-DP.DG.Creators.dgml*
Undo
Show Related
Layout
Share
Test/DP.DG.Implicit/Creators/ImplicitDpGraphCreator.cs
ImplicitDpGraphCreator
Test/DP.DG.Implicit/Creators/ImplicitActionsCreator.cs
ImplicitActionsCreator
Test/DP.DG/Creators/SimpleImplicitDpGraphCreator.cs
SimpleImplicitDpGraphCreator
DIR.Test.DP.DG.Implicit
Test/DP.DG.Implicit/ActivityService.cs

2. Significance of D_{imp}

Project	No. of implicit dependencies		Precision
	Original	Significant	
A	180	138	76.67%
B	112	103	91.96%
C	257	203	78.99%
D	634	576	90.85%

Dependencies on (and between) configuration files

Redirections between MVC controllers

JavaScript code calling a C# web service

Correct pairings of View and code-behind files

Transitive dependencies through interfaces

Further work

Distinguishing interactions by activities

- Significance of interactions differ for different activities
- Code study, debugging, testing, refactoring

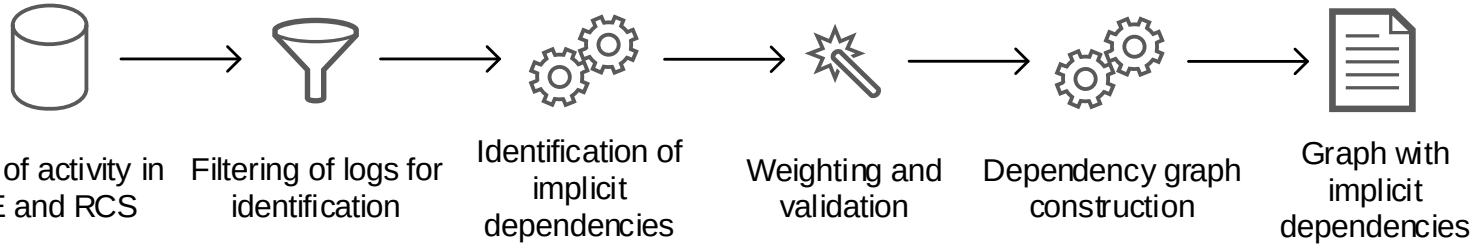
Evaluation

- Monitoring of ~20 professional web developers
- Source code components, not just files

Dynamic typing and inter-language support?

Software Developer Activity as a Source for Identifying Hidden Source Code Dependencies

Developer *implicitly* reveals *potential* dependencies



Independent from programming language

Task-related connections in source code

Further evaluation

PerConIK
Personalized Conveying of Information and Knowledge

martin_konopka@stuba.sk, maria.bielikova@stuba.sk

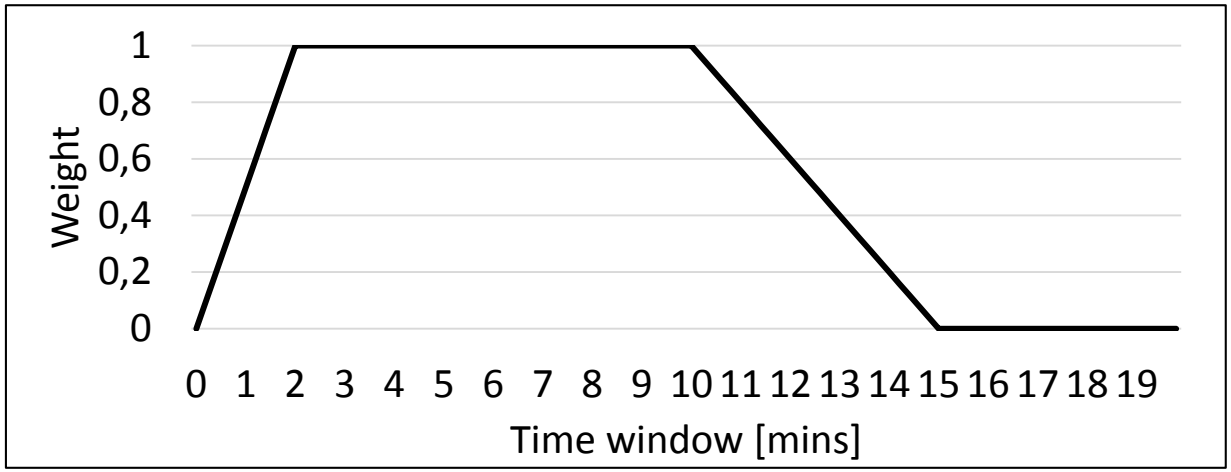
3. Weighting and validation

Determining significance for each kind differently

$$weight\colon D_{imp} \rightarrow \langle 0,1 \rangle$$

Time-related $D_{imp,time}$

- Dwell time in target component of navigation



3. Weighting and validation (cont.)

Content-related $D_{imp,content}$

- Based on contents of code fragment
- We chose weight each $d_{imp,content}$ with 1 (full significance)

Commit-related $D_{imp,commit}$

- Number of committed files

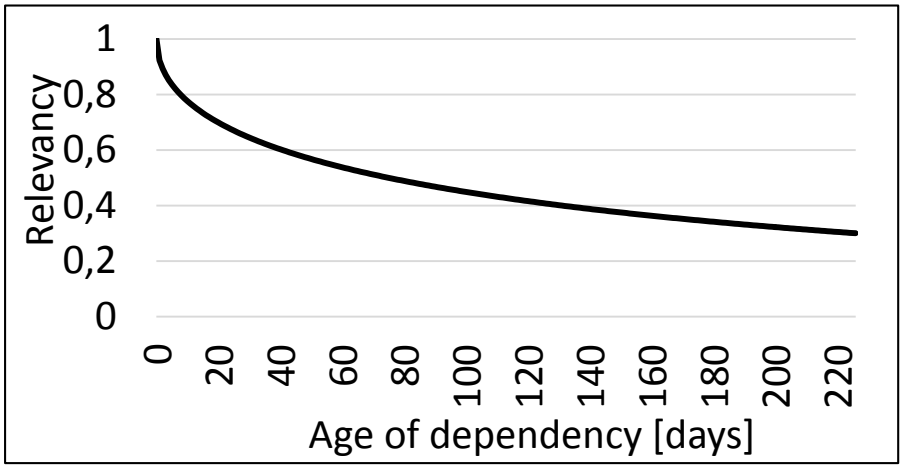
$$weight\left(d_{imp,commit}\right) = \frac{1}{count}$$

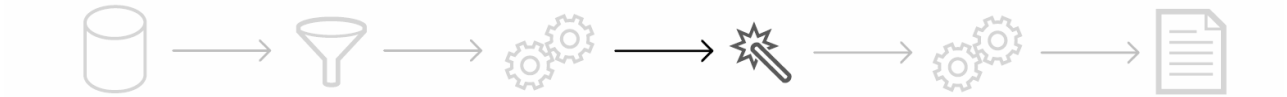
3. Weighting and validation

For explicit dependencies – yes/no from code

Decay of potential dependencies

$$validity_{imp}: D_{imp} \times T \rightarrow \langle 0,1 \rangle$$





Data properties

Source code metrics

- LOC, MI, CC

Number of explicit dependencies in graphs

- Code Map functionality in Microsoft Visual Studio

Number of interaction event logs (not filtered)

Project	LOC	MI	CC	$\lvert E_{exp} \rvert$	No. of interactions
A	2,402	82	1,285	232	15,215
B	4,384	74	2,164	274	8,584
C	4,993	81	3,213	528	24,213
D	5,342	81	1,839	270	55,877
E	3,721	81	1,779	189	48,717

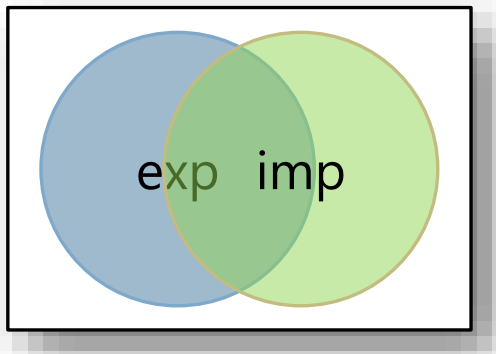
E1: Implicit vs explicit dependencies

1. Explicit dependencies among implicit ones

$$\frac{|E_{exp} \cap E_{imp}|}{|E_{imp}|}$$

2. Explicit dependencies identified from interactions

$$\frac{|E_{exp} \cap E_{imp}|}{|E_{exp}|}$$



E1: Implicit vs explicit - Results

Project	E_{imp} threshold	$\lvert E_{exp} \cap E_{imp} \rvert$	$\frac{\lvert E_{exp} \cap E_{imp} \rvert}{\lvert E_{imp} \rvert}$	$\frac{\lvert E_{exp} \cap E_{imp} \rvert}{\lvert E_{exp} \rvert}$
A	1	173	40.90%	74.57%
	2	108	54.00%	46.55%
B	1	140	41.30%	51.09%
	2	70	49.65%	25.55%
C	1	210	40.08%	39.77%
	2	120	53.57%	22,73%
D	1	191	34.35%	70.74%
	2	123	43.16%	45.56%
E	1	149	32.11%	78.84%
	2	108	43.90%	57.14%