IOWA STATE UNIVERSITY
Department of Computer Science

LSS
Laboratory for Software Safety

# The Role of Similarity in Detecting Feature Interaction in Software Product Lines

Zahra Khoshmanesh
Iowa State University
Dept. of Computer Science
zkh@iastate.edu

Robyn R. Lutz
Iowa State University
Dept. of Computer Science
rlutz@iastate.edu

2018 IEEE International Symposium on Software Reliability Engineering Workshops

# Outline

- **Introduction**
- Related Work
- Description Of Approach
- Results
- Conclusion and Future Work

# Software Product Line (SPL)

Family of software products that *share* common features as well as other optional or alternative features that it is worthwhile to study those products together as a *set*.[1]

[1] D. L. Parnas, "On the design and development of program families," IEEE Transactions on software engineering, no. 1, pp. 1–9, 1976.

# Feature-Oriented Domain Analysis (FODA)

- First *graphical* representation of a SPL.
- **Feature** : "a prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems,"[1]

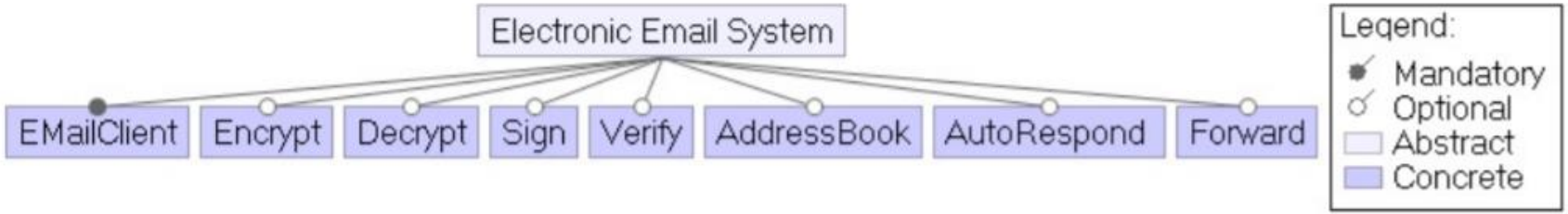



**Figure 1**: FODA representation of a Software product line case study

[1] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, Tech. Rep., 1990.

# Software Aging in SPL

- SPL evolve over time *as new features are added*.[1]
- Performance 
  Failure rate 
- One Possible Reason : *Feature Interaction*

[1] S. Johnsson and J. Bosch, “Quantifying software product line ageing, "Software Product Lines: Economics, Architectures, and Implications, 2000.

# Feature Interaction

- Integrate two or more features from a feature model to produce a new product, but they do not work together as intended (*Behavioral Interaction*) .
- Example :
  - Email Client + Encrypt/Decrypt ✓
  - Email Client + Forward ✓
  - Email Client + Encrypt/Decrypt + Forward ✗
- Violates a security property of system : E-mail content shall be kept encrypted if received email was encrypted

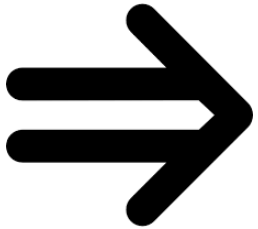
# Detection of Feature Interaction

- Difficult
- Currently depends primarily on *Testing*
  - Costly
  - Fails to exploit developers' knowledge of prior feature interactions that exist in the product line(PL)
  - Late

# Our Idea to improve detection of feature Interaction in a new Product in a SPL

knowledge of prior feature interactions in a PL

Similarity measures



detect earlier,
in the design phase.

# Two Goals

Understand whether information about features' structural elements suffices to detect potential new Feature Interaction.

Investigate whether similarity measures can help achieve this detection.

# Research Questions

## RQ1

How effectively can we measure the similarity between a new feature and existing features using structural similarity measures?

## RQ2

To what extent does a high similarity measure between a new and an existing feature, in the context of a known feature interaction, detect possible feature interactions in a new product?

# Case study: Electronic Email System[Hall(2005), Apel(2011)]

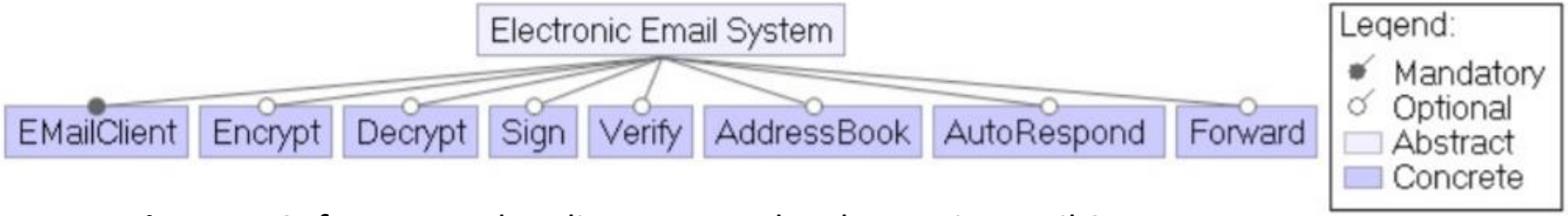



**Figure 1:** Software product line case study: Electronic Email System

**Addressbook** : manage e-mail contacts
**Autoresponder**: respond to e-mails
**Email Client** or **Base** : basic e-mail client
**Decrypt** : decrypt incoming e-mails
**Encrypt** : encrypt outgoing e-mails
**Forward** : forward incoming e-mails
**Sign** : sign outgoing e-mails
**Verify** : verify e-mail signatures

# Outline

- Introduction
- **Related Work**
- Description Of Approach
- Results
- Conclusion and Future Work

# Related Work

- **Johnsson and Bosch(2000)** proposed an approach to *quantify the aging* of a software architecture for a software product line.

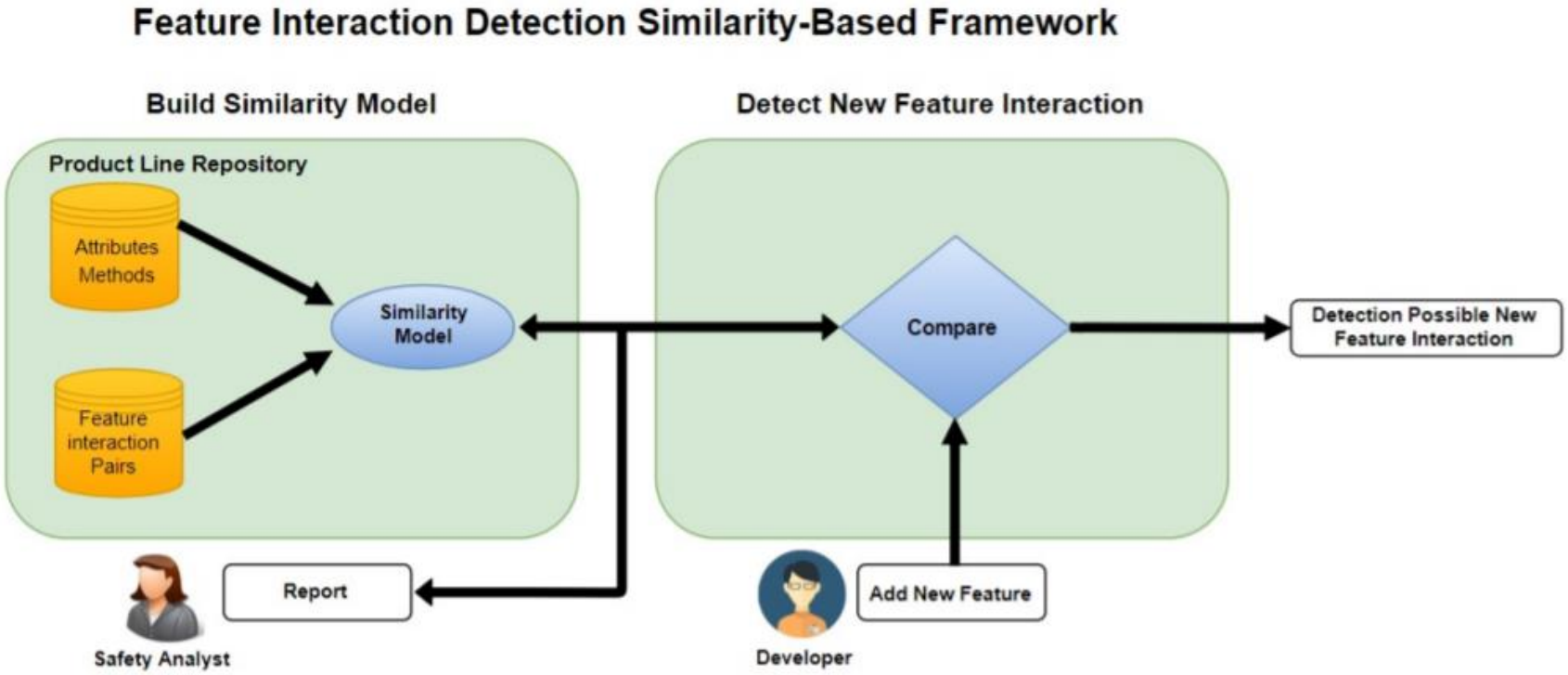
- **Cotroneo, Natella, and Pietrantuono(2013)** investigated the *relation* between *software aging* and *software metrics*.

- **Al-Hajjaji, et al.(2014)** proposed a *similarity-based prioritization* that increases coverage of *SPL test cases* to detect errors in reasonable time.

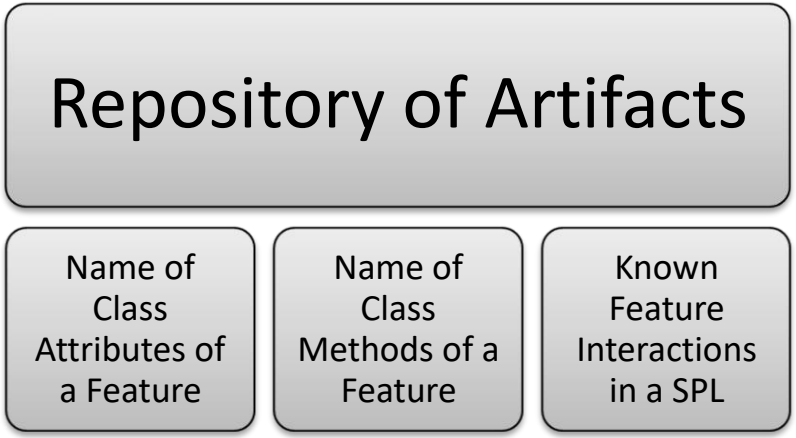
# Outline

- Introduction
- Related Work
- **Description Of Approach**
- Results
- Conclusion and Future Work

# Our proposed Similarity Framework



# Artifacts We Used




| Feature Name | Attribute name | Method name |
|---|---|---|
| Decrypt | private key | incoming |
| Sign | private key, signed, signkey | outgoing, sign, printmail, issigned |

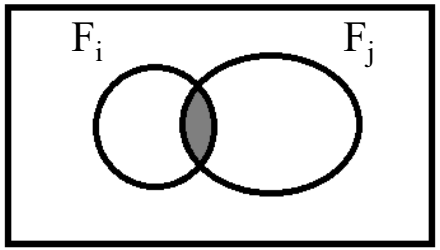
# Similarity measures

## Jaccard

$$J : F \times F \longrightarrow [0,1] \qquad (1)$$

$$J(Fi, Fj) = \frac{|Fi \cap Fj|}{|Fi \cup Fj|}, \quad \text{Where} \quad Fi, Fj \in F$$

S




**1**: Two features are identical

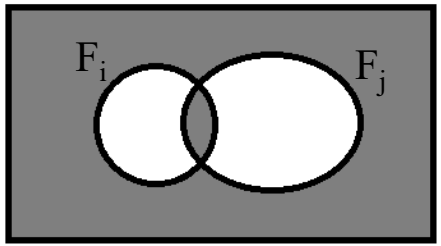
## Hamming

$$H : F \times F \longrightarrow [0,1] \qquad (2)$$

$$H(Fi, Fj) = \frac{|Fi \cap Fj| + |(S \backslash Fi) \cap (S \backslash Fj)|}{|S|},$$

$$\text{Where} \quad Fi, Fj \in F, S = \bigcup_{i \in I} Fi, I = \{1, ..., n\}$$

S




**0**: Two features are totally different

# Jaccard/Hamming Calculation

## Jaccard

$$F1 = \{a, b, c, d\}\ ,\ F2 = \{a, c, f\}$$
$$|F1 \cap F2| = |\{a, c\}| = 2$$
$$J(F1, F2) = \frac{|F1 \cap F2|}{|F1 \cup F2|} = \frac{2}{5} = 0.4$$

## Hamming

$$F1 = \{a, b, c, d\}\ ,\ F2 = \{a, c, f\}, F3 = \{b, e\}$$
$$S = \bigcup_{i \in I} Fi = F1 \cup F2 \cup F3 = \{a, b, c, d, e, f\}$$
$$|F1 \cap F2| = |\{a, c\}| = 2$$
$$|(S \backslash F1) \cap (S \backslash F2)| = |\{e\}| = 1$$
$$H(F1, F2) = \frac{|F1 \cap F2| + |(S \backslash F1) \cap (S \backslash F2)|}{|S|} = \frac{2+1}{6} = 0.5$$

# Known feature interactions from the original sources

[Hall(2005), Apel(2011)]

| Feature Id | Features Involved |
|---|---|
| 0 | Decrypt , Forward |
| 1 | Addressbook ,Encrypt |
| 3 | Sign ,Verify |
| 4 | Sign , Forward |
| 6 | Encrypt , Decrypt |
| 7 | Encrypt , Verify |
| 8 | Encrypt , Autoresponder |
| 9 | Encrypt , Forward |
| 11 | Decrypt , Autoresponder |
| 13 | Autoresponder , Forward |
| 27 | Verify , Forward |

# Predicting a New Feature Interaction

| *Known Feature Interaction* | **Encrypt, Autoresponder (8)** | | |
|---|---|---|---|
| *Possible new features* | Addressbook, Decrypt, Forward, Sign, Verify | | |
| | ***Encrypt*** | | ***Autoresponder*** |
| ***Similar features by Jaccard*** | Verify | | Decrypt |
| *Jaccard Detection* | Verify-Autoresponder | | Encrypt-Decrypt |
| *True Detection* | x | | ✓ |
| ***Similar feature by Hamming*** | Decrypt Verify | | Decrypt |
| *Hamming Detection* | (Decrypt – Autoresponder) | (Verify– Autoresponder) | Encrypt-Decrypt |
| *True Detection* | ✓ | x | ✓ |

# Predicting a New Feature Interaction

| *Known Feature Interaction* | **Encrypt, Autoresponder (8)** | |
|---|---|---|
| *Possible new features* | Addressbook, Decrypt, Forward, Sign, Verify | |
| | ***Encrypt*** | ***Autoresponder*** |
| ***Similar features by Jaccard*** | Verify | Decrypt |

# Predicting a New Feature Interaction

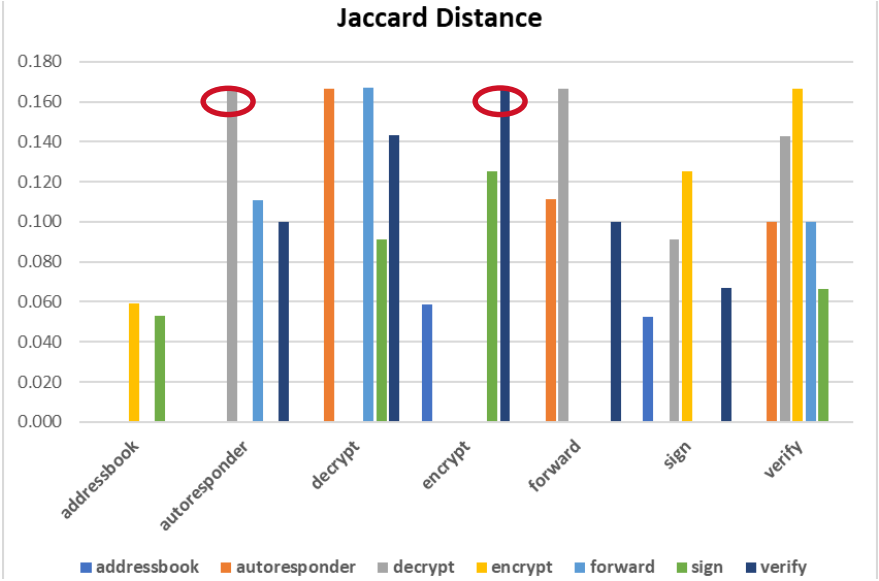
| *Known Feature Interaction* | **Encrypt, Autoresponder (8)** | |
|---|---|---|
| *Possible new features* | Addressbook, Decrypt, Forward, Sign, Verify | |
| | ***Encrypt*** | ***Autoresponder*** |
| ***Similar features by Jaccard*** | Verify | Decrypt |

**Jaccard Distance ([0-1] ;1:same 0:totally different)**

| | addressbook | autoresponder | decrypt | encrypt | forward | sign | verify |
|---|---|---|---|---|---|---|---|
| addressbook | | 0.000 | 0.000 | 0.059 | 0.000 | 0.053 | 0.000 |
| autoresponder | 0.000 | | 0.167 | 0.000 | 0.111 | 0.000 | 0.100 |
| decrypt | 0.000 | 0.167 | | 0.000 | 0.167 | 0.091 | 0.143 |
| encrypt | 0.059 | 0.000 | 0.000 | | 0.000 | 0.125 | 0.167 |
| forward | 0.000 | 0.111 | 0.167 | 0.000 | | 0.000 | 0.100 |
| sign | 0.053 | 0.000 | 0.091 | 0.125 | 0.000 | | 0.067 |
| verify | 0.000 | 0.100 | 0.143 | 0.167 | 0.100 | 0.067 | |

# Predicting a New Feature Interaction

| *Known Feature Interaction* | **Encrypt, Autoresponder (8)** | |
|---|---|---|
| *Possible new features* | Addressbook, Decrypt, Forward, Sign, Verify | |
| | ***Encrypt*** | ***Autoresponder*** |
| ***Similar features by Jaccard*** | Verify | Decrypt |



# Predicting a New Feature Interaction

| *Known Feature Interaction* | **Encrypt, Autoresponder (8)** | |
|---|---|---|
| *Possible new features* | Addressbook, Decrypt, Forward, Sign, Verify | |
| | ***Encrypt*** | ***Autoresponder*** |
| ***Similar features by Jaccard*** | Verify | Decrypt |
| *Jaccard Detection* | Verify-Autoresponder | Encrypt-Decrypt |
| *True Detection* | ✗ | ✓ |

| Feature Id | Features Involved |
|---|---|
| 0 | Decrypt , Forward |
| 1 | Addressbook ,Encrypt |
| 3 | Sign ,Verify |
| 4 | Sign , Forward |
| 6 | Encrypt , Decrypt |
| 7 | Encrypt , Verify |
| 8 | Encrypt , Autoresponder |
| 9 | Encrypt , Forward |
| 11 | Decrypt , Autoresponder |
| 13 | Autoresponder , Forward |
| 27 | Verify , Forward |

# Outline

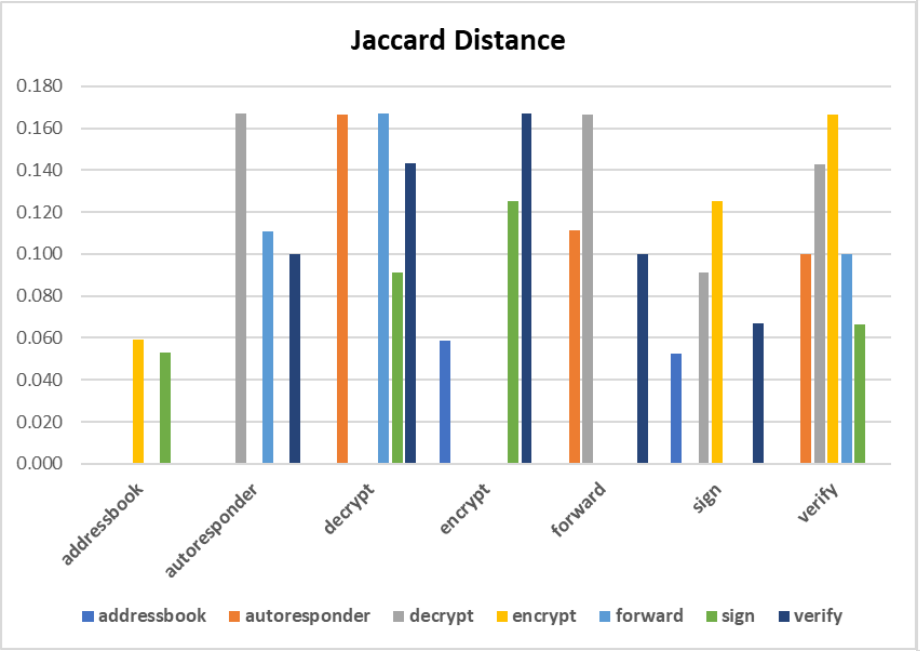
- Introduction
- Related Work
- Description Of Approach
- **Results**
- Conclusion and Future Work

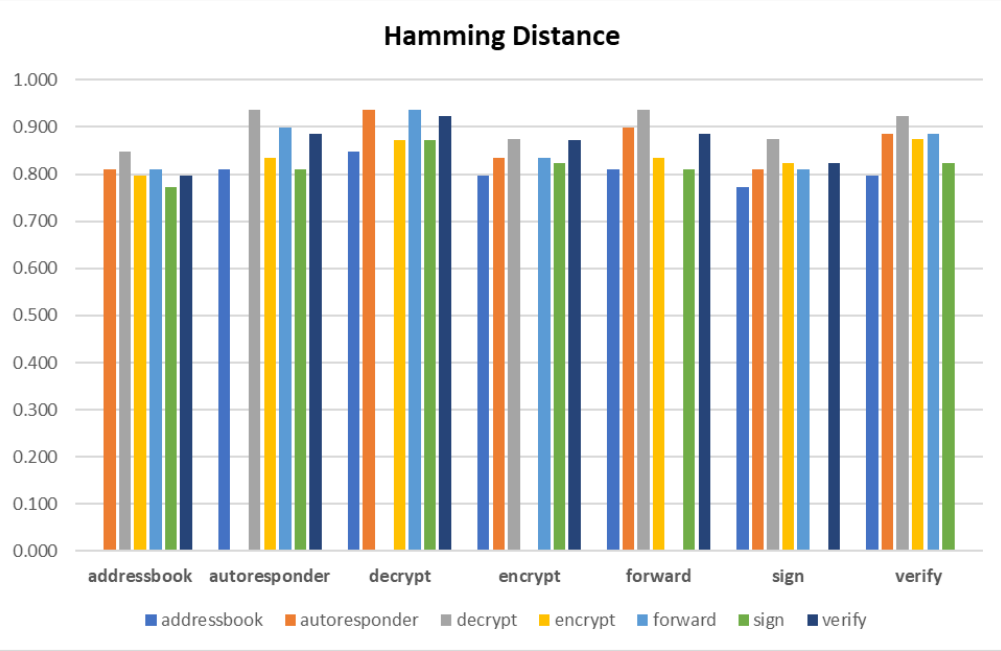
# RQ1: How effectively can we measure the similarity between a new feature and existing features using structural similarity measures?

**Yes,**
We can effectively measure the similarity of two features.

| | Jaccard Distance ([0-1] ; 1:same 0:totally different) | | | | | | |
|---|---|---|---|---|---|---|---|
| | | addressbook | autoresponder | decrypt | encrypt | forward | sign | verify |
| 3 | addressbook | | 0.000 | 0.000 | 0.059 | 0.000 | 0.053 | 0.000 |
| 4 | autoresponder | 0.000 | | 0.167 | 0.000 | 0.111 | 0.000 | 0.100 |
| 5 | decrypt | 0.000 | 0.167 | | 0.000 | 0.167 | 0.091 | 0.143 |
| 6 | encrypt | 0.059 | 0.000 | 0.000 | | 0.000 | 0.125 | 0.167 |
| 7 | forward | 0.000 | 0.111 | 0.167 | 0.000 | | 0.000 | 0.100 |
| 8 | sign | 0.053 | 0.000 | 0.091 | 0.125 | 0.000 | | 0.067 |
| 9 | verify | 0.000 | 0.100 | 0.143 | 0.167 | 0.100 | 0.067 | |
| 10 | | | | | | | | |
| 11 | | | | | | | | |
| 12 | Hamming Distance ([0-1] ; 1:same 0:totally different) | | | | | | | |
| 13 | | addressbook | autoresponder | decrypt | encrypt | forward | sign | verify |
| 14 | addressbook | | 0.810 | 0.848 | 0.797 | 0.810 | 0.772 | 0.797 |
| 15 | autoresponder | 0.810 | | 0.937 | 0.835 | 0.899 | 0.810 | 0.886 |
| 16 | decrypt | 0.848 | 0.937 | | 0.873 | 0.937 | 0.873 | 0.924 |
| 17 | encrypt | 0.797 | 0.835 | 0.873 | | 0.835 | 0.823 | 0.873 |
| 18 | forward | 0.810 | 0.899 | 0.937 | 0.835 | | 0.810 | 0.886 |
| 19 | sign | 0.772 | 0.810 | 0.873 | 0.823 | 0.810 | | 0.823 |
| 20 | verify | 0.797 | 0.886 | 0.924 | 0.873 | 0.886 | 0.823 | |
| 21 | | | | | | | | |

# Jaccard/Hamming Distance between features in the Electronic Mail System



# RQ2 :To what extent does a high similarity measure between a new and an existing feature, in the context of a known feature interaction, detect possible feature interactions in a new product?

| | True detection | False detection | Accuracy |
|---|---|---|---|
| **Jaccard** | 16 | 7 | 70% |
| **Hamming** | 19 | 7 | 73% |

# Summary of feature interaction detection using Jaccard and Hamming distance

| FI ID | Detected by Jaccard | Detected by Hamming |
|---|---|---|
| 0 | ✓ | ✓ |
| 1 | X | X |
| 3 | ✓ | X |
| 4 | ✓ | ✓ |
| 6 | ✓ | ✓ |
| 7 | ✓ | ✓ |
| 8 | ✓ | ✓ |
| 9 | ✓ | ✓ |
| 11 | ✓ | ✓ |
| 13 | ✓ | ✓ |
| 27 | ✓ | ✓ |
| Total | 10 | 9 |

**1** Addressbook ,Encrypt

**Why Not** detected?
Addressbook does not have the highest similarity value of any of the features in the SPL

# Outline

- Introduction
- Related Work
- Description Of Approach
- Results
- **Conclusion and Future Work**

# Conclusion

- The heuristic behind this study :
  - similar features often behave in the same way
- Our results :
  - using similarity measures between features in a software product line can help us detect possible feature interactions at the design stage.

# Future Work

Explore more case studies ,

- Additional structural artifacts
- Improved similarity metric


# Thank you!

Supported in part by NSF grant 1513717

# Reference


- [1] D. L. Parnas, “On the design and development of program families,” IEEE Transactions on software engineering, no. 1, pp. 1–9, 1976.
- [2] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study,” Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, Tech. Rep., 1990.
- [3] S. Johnsson and J. Bosch, “Quantifying software product line ageing, "Software Product Lines: Economics, Architectures, and Implications, 2000.
- [4]R. J. Hall, “Fundamental nonmodularity in electronic mail,” Automated Software Engineering, vol. 12, no. 1, pp. 41–79, 2005.
- [5] S. Apel, H. Speidel, P. Wendler, A. Von Rhein, and D. Beyer, “Detection of feature interactions using feature-aware verification,” in Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering. IEEE Computer Society, 2011, pp. 372–375.
- [6] D. Cotroneo, R. Natella, and R. Pietrantuono, “Predicting aging-related bugs using software complexity metrics,” Performance Evaluation, vol. 70, no. 3, pp. 163–178, 2013.
- [7] M. Al-Hajjaji, T. Th¨um, J. Meinicke, M. Lochau, and G. Saake, “Similarity-based prioritization in software product-line testing,” in Proceedings of the 18th International Software Product Line Conference-Volume 1. ACM, 2014, pp. 197–206.