

CISC 322

Software Architecture



Lecture 13:

Reflexion Models and Source Sticky Notes

Emad Shihab

Paper by: Ahmed E. Hassan and Richard C. Holt

Slides adapted from Ahmed E. Hassan

Midterm

Group presentations

■ Tuesday

- Team Phoenix
- Fully Optimized eXperience
- Fox Bytes
- The Firey Foxes

■ Wednesday

- Volpe Inferno
- The Flaming Pandas

■ Friday

- The Fighting Mongooses

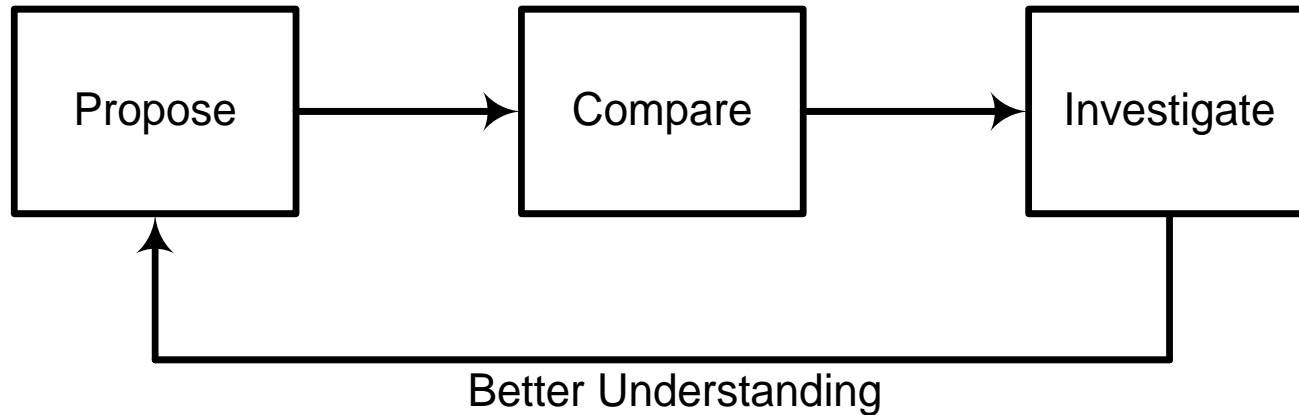
Introduction

- Software understanding tasks represent 50-90% of maintenance efforts
- Good documentation can help, but rarely available
- Some developers resort to code browsing, but that is limited and does not scale
- Propose to speedup understanding using knowledge from **historical modification records**

Source Sticky Notes

- Attach change details to dependencies between software entities
- Determine the affect of a change on a software's dependency graph and attach change details to edges of the dependency
- Provide insight to developers about reasons for that dependencies

Architecture Understanding Process



- *Propose* conceptual architecture
- *Compare* conceptual with concrete architecture
- *Investigate* gaps

Propose - Conceptual Architecture

- Developers propose a conceptual architecture based on:
 - Reference architecture
 - System documentation
 - Developer experience with similar systems
 - Talking to senior developers and domain experts

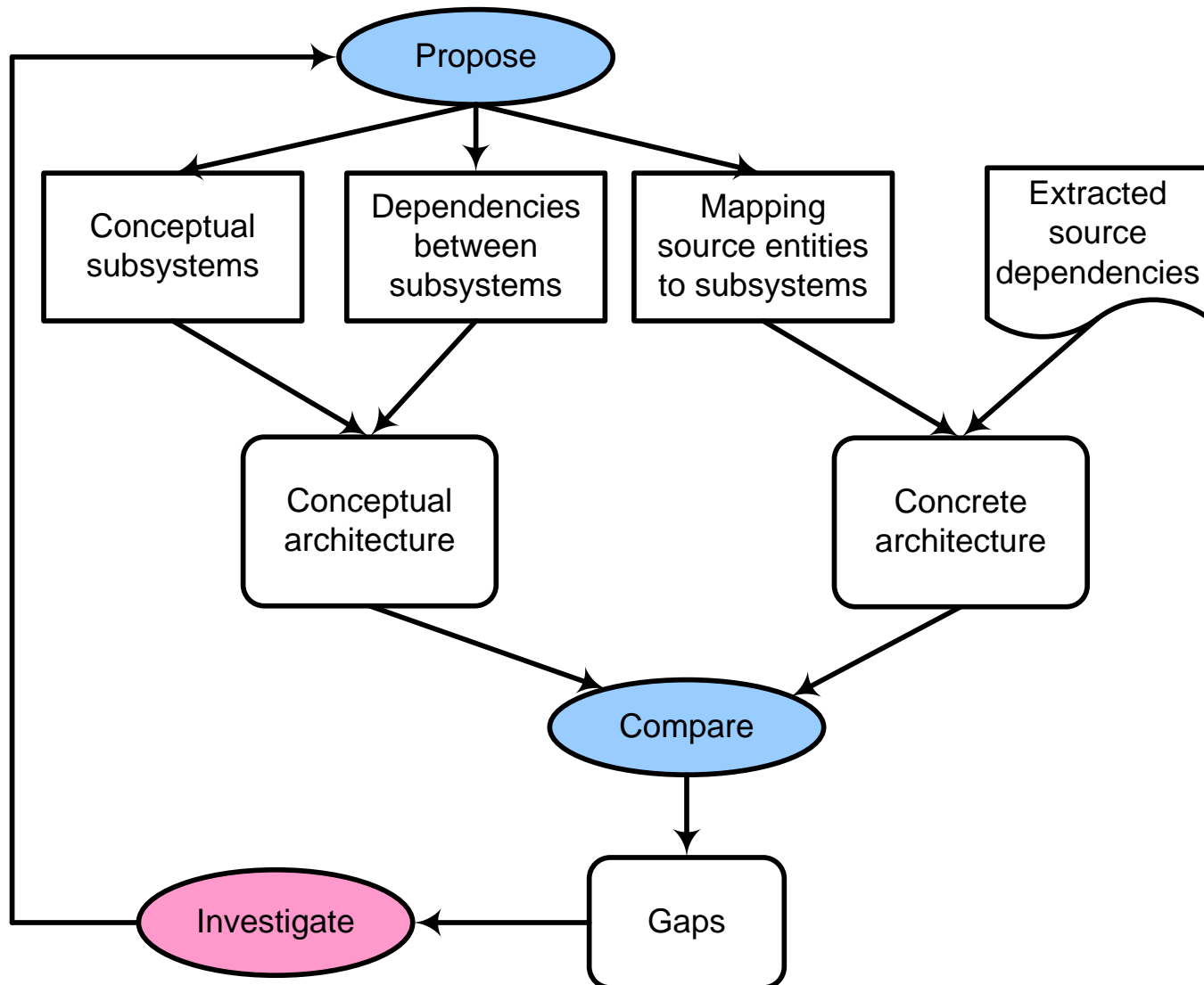
Conceptual and Concrete Mismatch

- However, in reality the concrete architecture is (almost) always different
- Need to not only discover differences, but also uncover the rationale

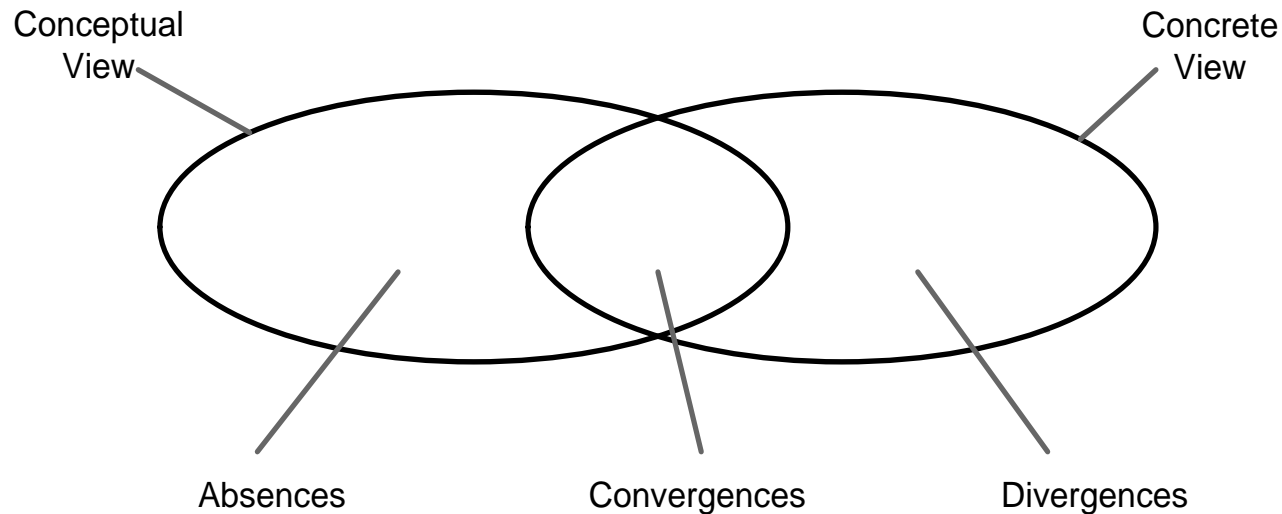
Uncovering the Rationale for the Differences

- Uncovering the rationale is challenging
 - A senior developer
 - may be too busy
 - may not recall the rationale for such dependency
 - may no longer work on the software system
 - The software
 - may have been bought from another company
 - may have its maintenance out-sourced
- Developers must spend hours/days to uncover the rationale. The rationale may be:
 - Justified due to, e.g., optimizations or code reuse; or
 - Not justified due to, e.g., developer ignorance or pressure to market.

Software Reflexion Framework



Investigating Gaps



- **Absences:** rarely occur in large systems
- **Convergences:** usually not a concern
- **Divergences:** must investigate dependencies

4Ws when Investigating Dependencies

Which?

- Which concrete source code entities are responsible for an unexpected dependency?

Who?

- Who introduced an unexpected dependency or removed a missing dependency?
- A gap due to a change made by
 - a **novice developer** may suggest that the developer is at fault and the change must be fixed
 - a **senior developer** with a well established record for producing high quality code may suggest that the change is correct

When?

- When was the unexpected dependency added or the missing dependency removed?
- A fix to a critical bug under a tight release schedule?
 - E.g. a few days/hours before a release
- Or is it is a justified dependency that we should expect

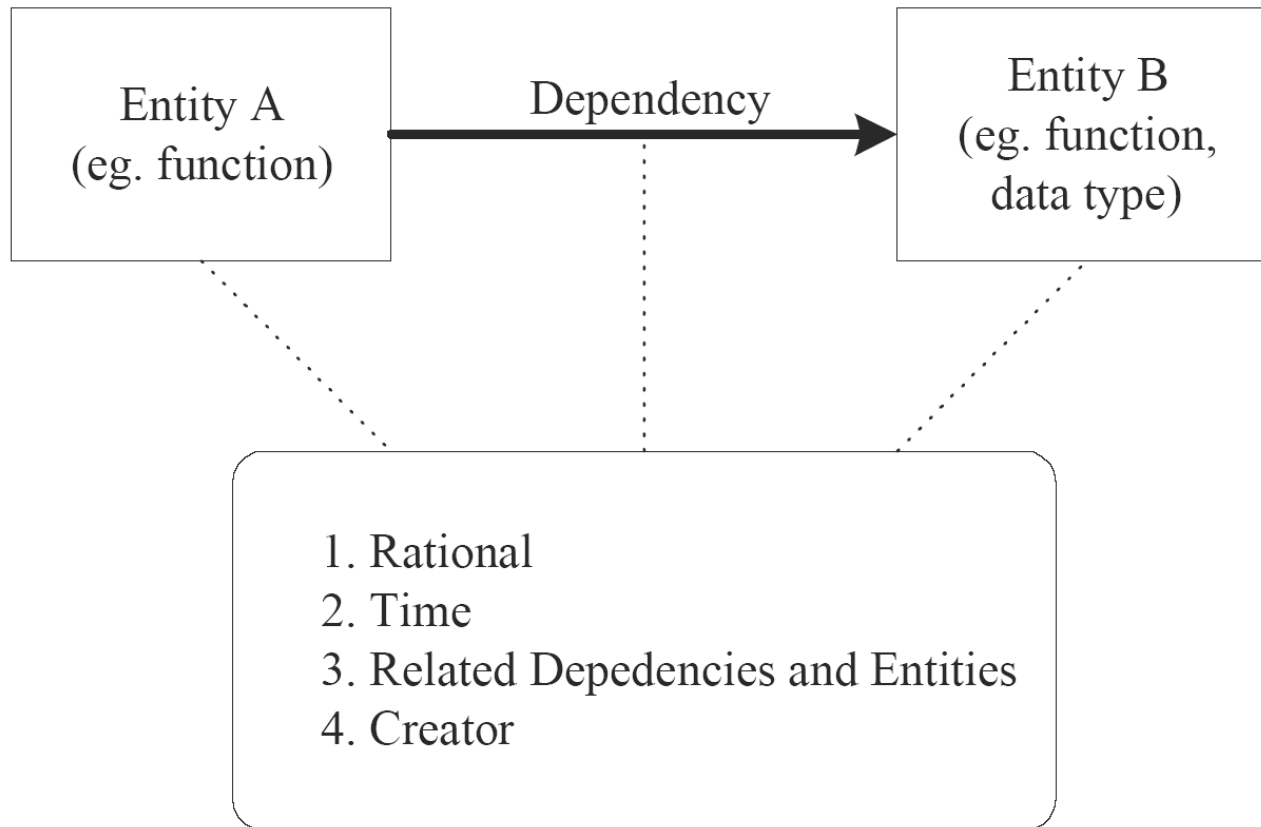
Why?

- Why was this unexpected dependency added or why was an expected dependency missing?
- A knowledge of the rationale is key in explaining the gaps

Dependency Investigation Questions (W4 Approach)

- **Which** low level code entity is responsible for the dependency?
 - Network (*SendData*) → Scheduler (*PrintToLog*)
- **Who** added/removed the dependency?
 - Junior vs. senior/experienced developer
- **When** was the dependency modified?
 - Late night / Just before release
- **Why** was the dependency added/removed?
 - The rationale!

Source *StickyNotes*



- We are interested in
 - Current and past dependencies

Source *StickyNotes*

- Static dependencies give only a current static view of the system – not enough detail!
- Need to extend static dependencies, but how?

Extending Code Dependencies

- Ask developers to fill StickyNotes for each change
 - Too time consuming and cumbersome
- Use software repositories to build these notes automatically
 - Historical information may be hard to process

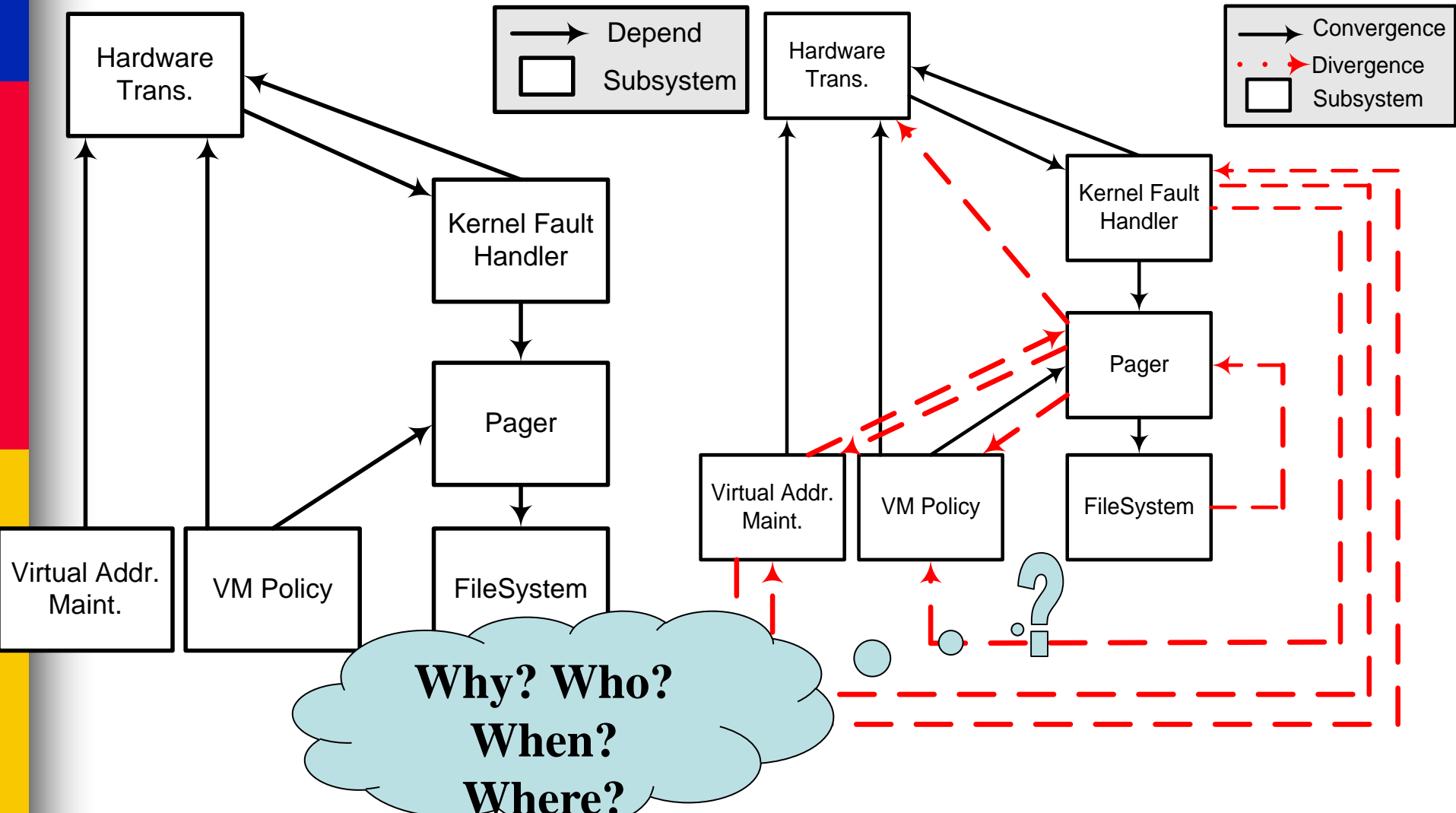
StickyNotes Recovery

- Map code changes to entities and dependencies instead of lines
- Two pass analysis of the source control repository data, to recover:
 - Record all entities defined throughout the lifetime of a project
 - Record all dependencies between these entities and attach source control meta-data

Case Study – NetBSD

- Large long lived system with hundreds of developers
- Case study used to demonstrate usefulness of the reflexion model:
 - Reuse prior results! 😊
 - Focus on investigating *gaps* to show the strength of our approach

NetBSD (VMC) Conceptual and Reflexion Model



Unexpected Dependencies

- Eight unexpected dependencies
- All except two dependencies existed since day one:
 - Virtual Address Maintenance → Pager

Which?	vm_map_entry_create (in src/sys/vm/Attic/vm_map.c) <i>depends on</i> pager_map (in /src/sys/uvm/uvm_pager.c)
Who?	cgd
When?	1993/04/09 15:54:59 Revision 1.2 of src/sys/vm/Attic/vm_map.c
Why?	<p>from sean eric fagan: it seems to keep the vm system from deadlocking the sy pr an in have to wait for a free page).</p> <div>Dependency added to avoid deadlocking under special circumstances</div>

Unexpected Dependencies

■ Pager → Hardware Translations

Which?	uvm_pagermapin (in src/sys/uvm/uvm_pager.c) <i>depends on</i> pmap_kenter_pgs (in src/sys/arch/arm26/arm26/Attic/pmap.c)
Who?	thorpej
When?	1999/05/24 23:30:44; Revision 1.17 of src/sys/uvm/uvm_pager.c
Why?	<p>Don't use pmap_kenter_pgs() for entering pager_map mappings. The pages are still owned by the object which is paging, and so the test for a kernel object in uvm used</p> <p>ins</p> <p>The vs</p> <p>pmap_kremove() inconsistency (which caused the busted kernel pmap statistics, and a cause of much locking hair on MP systems).</p>

Dependency added to fix a bug on multiple process systems

Unexpected Dependencies which existed in the past

- Two unexpected dependencies that were removed in the past:
 - Hardware Translation → VM Policy
 - File System → Virtual Address Maintenance

Which?	mfs_strategy (in.src/sys/ufs/mfs/mfs_vnops.c) <i>depends on</i> vm_map (in src/sys/vm/Attic/vm_map.h)
Who?	thorpej
When?	2000/05/19 20:42:21; Revision 1.23 of src/sys/ufs/mfs/mfs_vnops.c
Why?	<div>Dependency removed to fix a previous incorrect change</div> <div>thing</div>

StickyNotes Usage Patterns

- ***First note*** to understand the reason for unexpected dependencies
- ***Last note*** to study missing dependencies
- ***All notes*** when first and last notes do not have enough information to assist in understanding

Limitations

- Quality of comments and text entered by developers in the past
- In many open source projects, CVS comments are used for:
 - Communicating new features
 - Narrating the progress of a project

Summary

- Development history can help understand the current structure of a software system
- Traditional dependency graphs and program understanding models usually do not use historical information
- Proposed *StickyNotes* and presented a case study to show the strength of the approach