

Logic Online Inc.

RIPPLE-TRAC VISUALIZATION



©

Written by
Daniel B. Boucher Sr,
z/OS Solutions Architect,
Logic Online, Inc.

A Separation of Concerns (SoC) Tool

www.logiconlineinc.com

© 2011 Logic Online Inc ALL RIGHTS RESERVED.

This document contains proprietary information, protected by copyright. No part of this document may be reproduced or transmitted for any purpose other than the reader's personal use without the permission of Logic Online Inc.

WARRANTY

The information contained in this document is subject to change without notice. Logic Online makes no warranty of any kind with respect to this information. Logic Online shall not be liable for any direct, indirect, incidental, consequential, or other damage alleged in connection with the use of this information.

TRADEMARKS

All trademarks and registered trademarks used in this guide are property of their respective owners.

RIPPLE-TRAC is a registered trademark in the state of New Hampshire

Logic Online Inc.
1500A Lafayette Rd #170
Portsmouth, NH 03801
e-mail: dan.boucher@logiconlineinc.com
www.logiconlineinc.com



www.logiconlineinc.com is a resource for:



<http://www-03.ibm.com/systems/z/destinationz/>

Multi-dimensional separation of concerns is an approach to separation of concerns, supporting construction, evolution and integration of software. Its goals are to enable:

- Encapsulation of all kinds of concerns in a software system, simultaneously.
- Overlapping and interacting concerns.
- On-demand remodularization.

Separation of concerns is a concept that is at the core of software engineering. It refers to the ability to identify, encapsulate, and manipulate those parts of software that are relevant to a particular concern (concept, goal, purpose, etc.). Concerns are the primary motivation for organizing and decomposing software into manageable and comprehensible parts. Many kinds of concerns may be relevant to different developers in different roles, or at different stages of the software lifecycle. Appropriate separation of concerns has been hypothesized to reduce software complexity and improve comprehensibility; promote traceability; facilitate reuse, non-invasive adaptation, customization, and evolution; and simplify component integration.

The term *multi-dimensional separation of concerns* (MDSOC) refers to flexible and incremental separation, modularization, and integration of software artifacts based on any number of concerns. It overcomes limitations of existing mechanisms by permitting clean separation of multiple, potentially overlapping and interacting concerns simultaneously. MDSOC promotes reuse, improves comprehension, reduces the impact of change, eases maintenance and evolution, improves traceability, and opens the door to system refactoring and reengineering.

MDSOC summary:

Involves decomposition of software according to one or more dimensions of concern. A concern is any piece of interest or focus in a program. http://en.wikipedia.org/wiki/Separation_of_concerns

The separation allows:

- To allow people to work on individual pieces of the system in isolation;
- To facilitate reusability;
- To ensure the maintainability of a system;
- To add new features easily;
- To enable everyone to better understand the system;
- To allow support for multi-dimensional separation of concerns.

Remember, a dimension of concern is simply an approach to decomposing, organizing, and structuring software according to concerns of a particular kind. **RIPPLE-TRAC** falls into the realm of multi-dimensional separation of concerns.

RIPPLE-TRAC OFFERS A SOLUTION FOR DATA SILOS AND APPLICATION REDUNDANCY

Legacy systems have been tuned and streamlined over many years, and they most likely perform very well. Leveraging these existing assets in place, where possible, is the best approach to capitalizing on the investment and years of work spent perfecting them.

An **information silo** is a management system incapable of shared operations with other, related management systems. A bank's management system, for example, is considered a silo if it cannot exchange information with other related systems within its own organization, or with the management systems of its customers, vendors, or business partners. "Information silo" is a derogatory expression that is useful for describing this type of relationship.

The *silo effect* is a phrase that is currently popular in the business and organizational communities to describe a lack of communication and common goals between departments in an organization. It is the opposite of systems thinking in an organization. The silo effect gets its name from the farm storage silo, probably because there could be five silos right next to each other and if people were inside them they would not be able to communicate, since silos are tall, narrow buildings with no windows and are even supposed to be airtight.[WIKIPEDIA] http://en.wikipedia.org/wiki/Systems_thinking

Let's set the stage for discussion:

Analyzing the impact of software modifications in a geographically distributed world

Most companies would deal with such changes in the most natural and straightforward way: They'd have each of the teams analyze the impacts to their applications and then act accordingly. In this scenario, we'd probably end up with different teams simultaneously doing similar activities on the same products. But it's common for businesses to have multiple projects going on at once in different locations, in-house or outsourced, and all of them need to be updated to reflect the modifications.

Software modifications provide another fertile opportunity for defects to invade code.

When requirements are poorly defined or managed—or when they change without accurate tracking—serious defects can be introduced.

Software quality optimization:

One can't help but assume that this approach creates a great deal of waste as well as increased risk. What if different teams interpret the new requirements differently? And what if they end up with test cases that reflect different behaviors when they were supposed to be identical? Dealing with multiple projects without significantly increasing the risk requires a testing process where a single set of requirements can be mapped to multiple projects. It requires the ability to define test cases from which test scripts can be generated and/or derived. It requires the ability to share and reuse test cases among projects. It requires a test plan that, like the requirements, is not a frozen document or a hierarchical Web page. It's a live, dynamic, always-up-to-date asset that allows for the definition and sharing of test processes and strategies, business-level reporting against quality objectives, and—just as important—support for collaborative team activities such as reviews and approvals.[IBM]

The following schematic depicts what a typical corporation might look like.

As indicated above, these silos do not know about each other.

Each silo may be geolocated or be at one site.

All silos may belong to the same LPAR or each silo may belong to its own LPAR.
Each instance represents an application area like data warehouse, general ledger, payroll etc.
Each file system is made up of PDS's, DB2, IMS, JCL, VSAM, CICS etc

SILO1 has 3 instances or business areas
SILO1 has 3 file systems that are not shared
SILO1 shares the corporate file system with SILO2 and SILO3

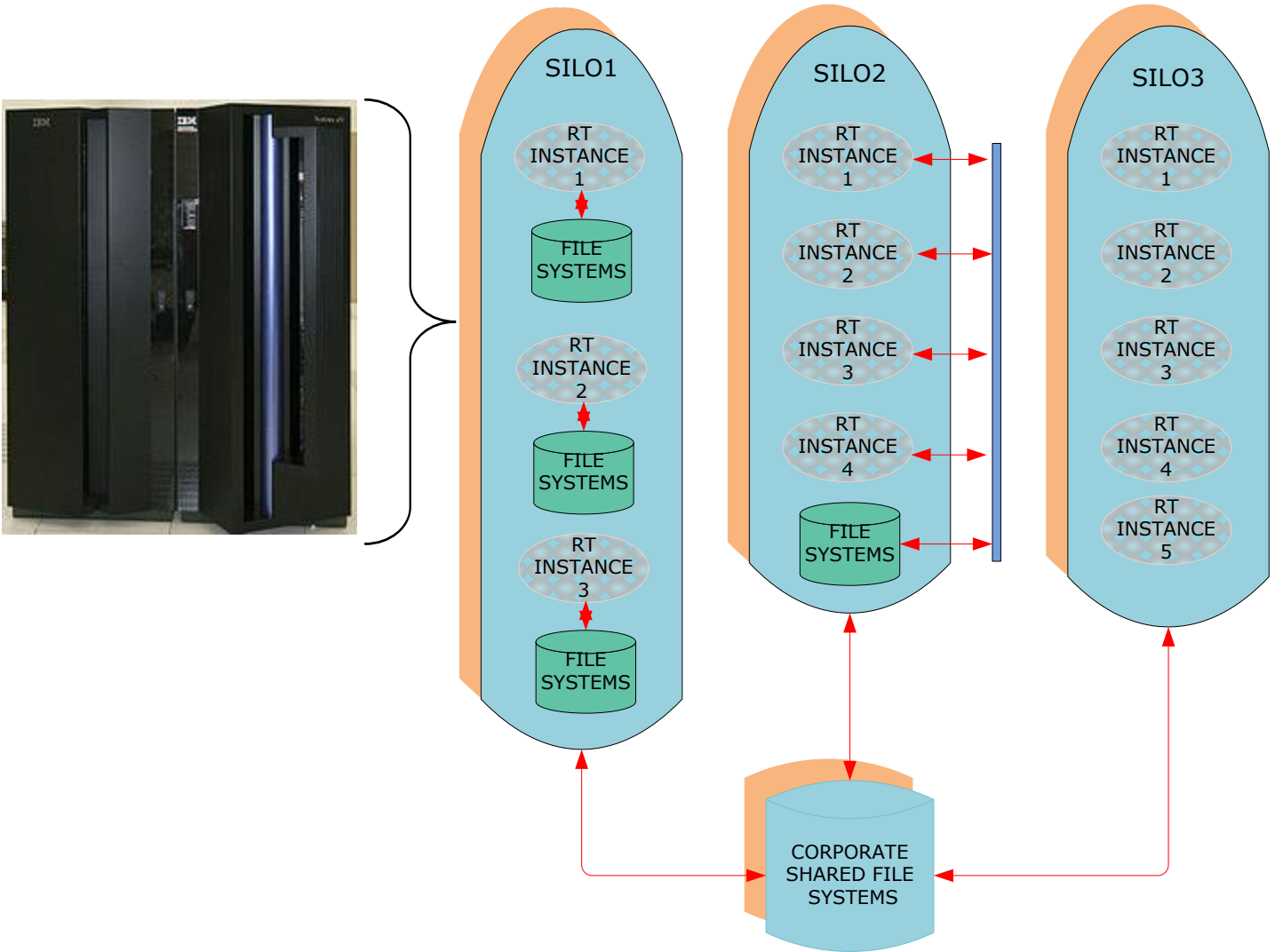
SILO2 has 4 instances or business areas
SILO2 has 1 file system that is shared by all instances in SILO2
SILO2 shares the corporate file system with SILO1 and SILO3

SILO3 has 5 instances or business areas
SILO3 does not have its own file system
SILO3 shares the corporate file system with SILO1 and SILO2

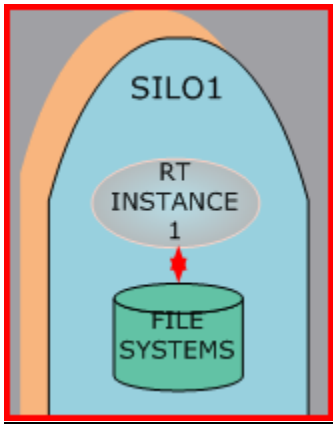
So, any combination within each silo is acceptable. For that matter, you could have all silos consolidated into one partition.

As you can see, these real world scenario's can get very complicated.

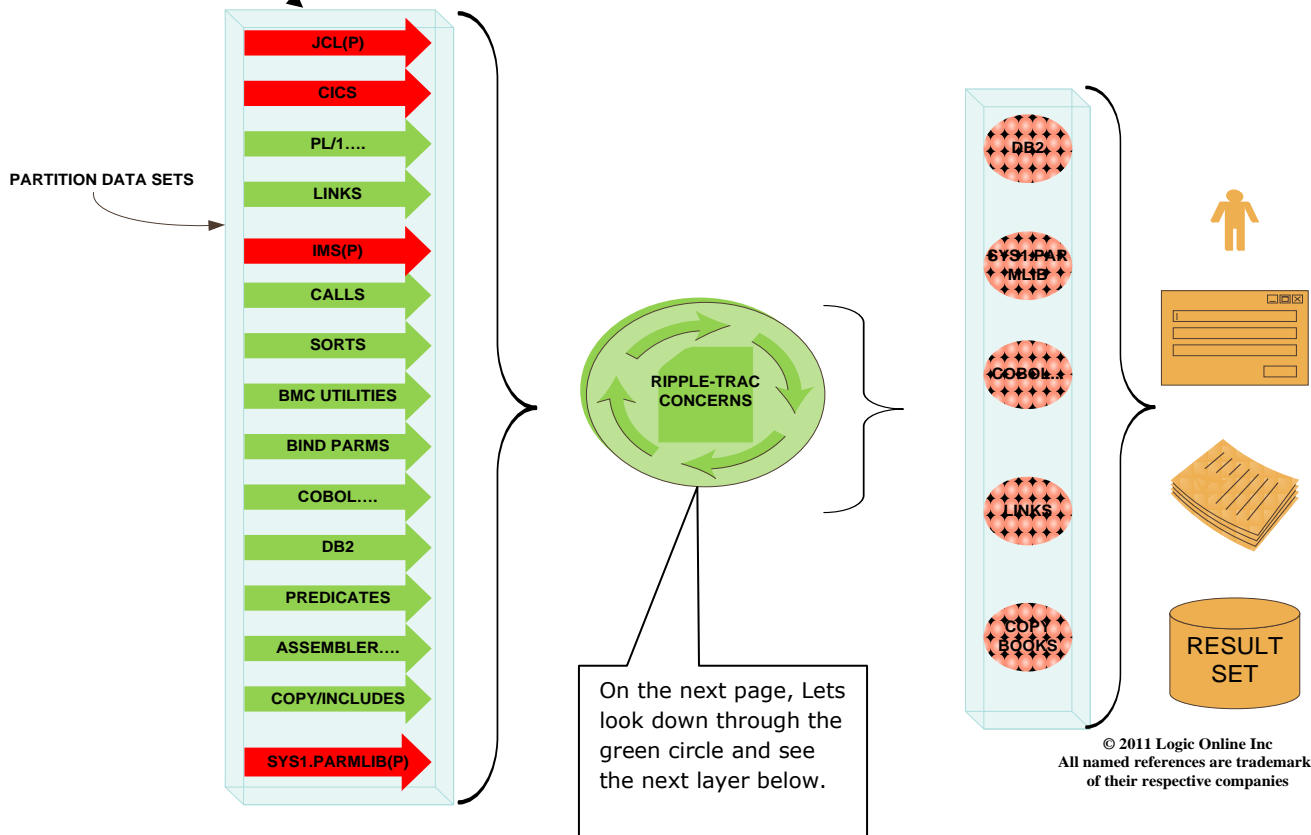
z/OS



The following schematic shows what **RIPPLE-TRAC** INSTANCE 1 IN SILO 1 might look like in an organization.



RIPPLE-TRAC VISUALIZATION

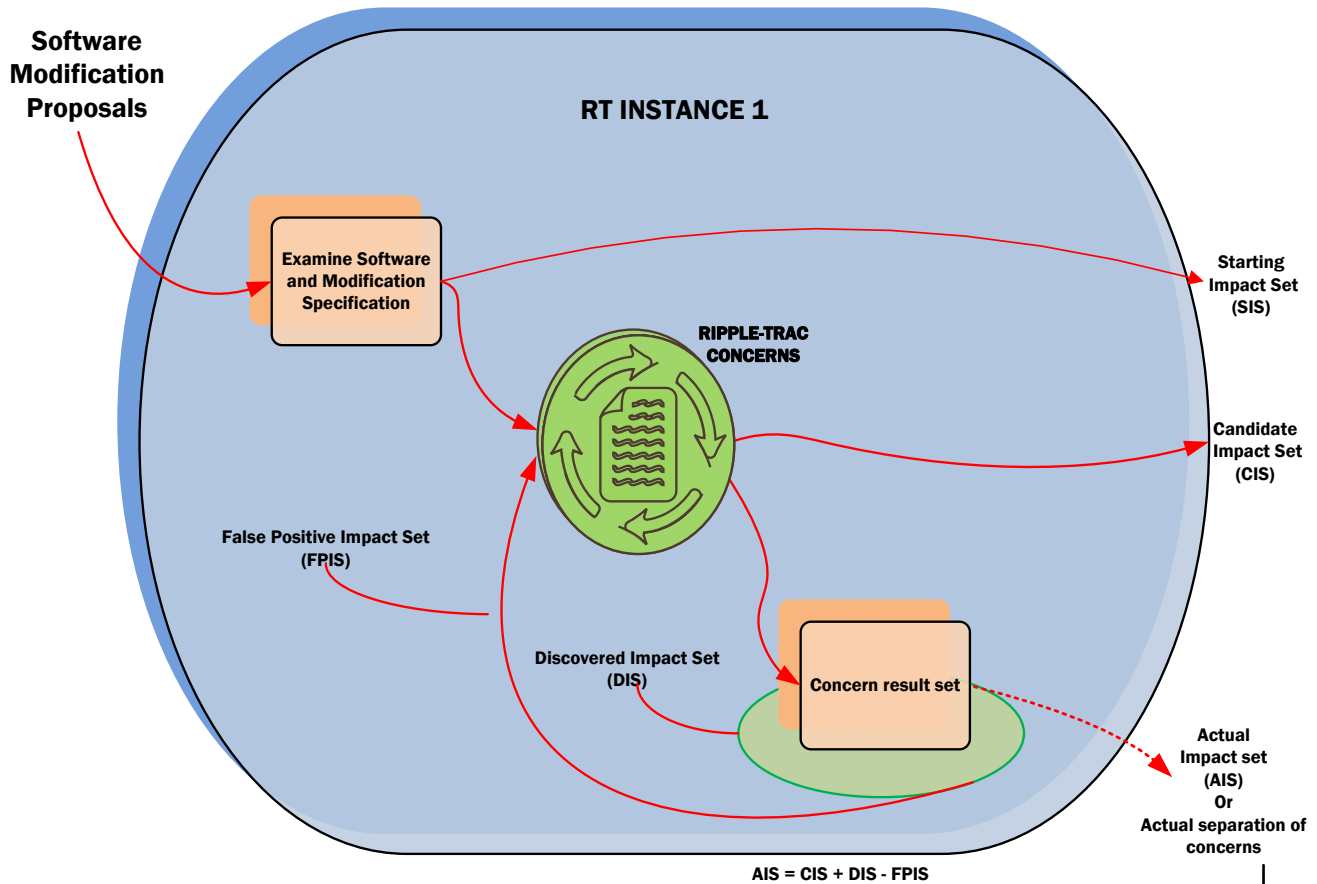


On the next page, Lets look down through the green circle and see the next layer below.

© 2011 Logic Online Inc
All named references are trademarks of their respective companies

NOTE: RED ARROWS INDICATES A WHITE PAPER AVAILABLE

RIPPLE-TRAC SEPARATION OF CONCERNS



The starting impact set (SIS) is the initial set of concerns thought to be affected by a modification. The SIS is normally determined while examining the modification specification. The candidate impact set (CIS) is the set of concerns estimated to be affected. The CIS is produced while conducting the impact analysis. The actual impact set (AIS) is the set of concerns that will actually be modified. The AIS is not necessarily unique, as a modification can be implemented in several ways.

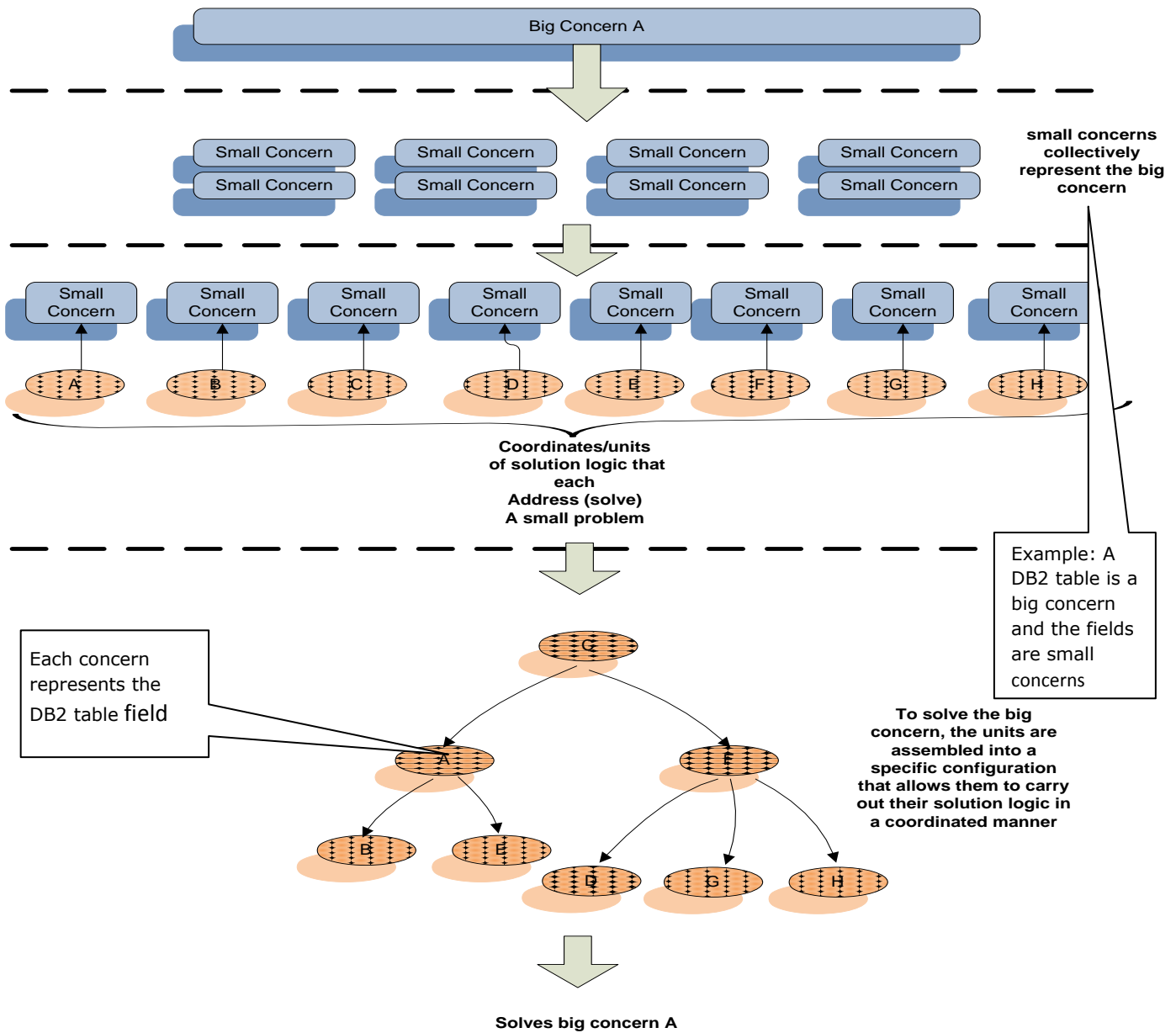
The impact analysis process is iterative and discovery in nature. While a modification is being performed, there are likely to be more impacts discovered. The discovered impact set (DIS) represents an under-estimate of impacts.

The false-positive impacts set (FPIS) signify the overestimate of impacts in the analysis. The CIS plus additions of the DIS and minus deletions of FBIS should represent the AIS. The accuracy (error) is obtained by adding the number of impacts in the DIS and FPIS then dividing them by the number of impacts in the CIS. This represents the number of errors (DIS and FBIS) divided by the estimate. The objective of the analysis is to have the Candidate Impact Set produced from tracing potential impacts as close to the Actual Impact Set as possible by identifying true impacts while eliminating false-positives.[1]

References

[1] Shawn A. Bohner and Denis Gracanic, "Software Impact Analysis in a Virtual Environment," NASA Goddard Software Engineering Workshop 2003

SEPARATION OF CONCERNS



The separation of concerns theory encourages us to break down a larger problem into multiple smaller problems (concerns). This gives us the opportunity to build corresponding pieces of solution logic, each of which solves a small problem (address an individual concern). These capabilities are part of units that are assembled into a composition through which they are coordinated to collectively solve the large problem.

END OF VISUALIZATION