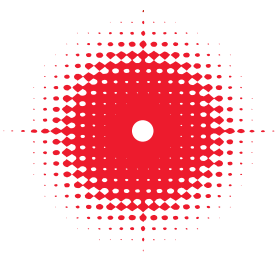
Anti-Patterns in Progress

Sean Overby

Senior Technologist, Solvepoint Corporation



SOLVEPOINT

SOLVEPOINT

Overview

- Patterns are common solutions to common programming problems.
- Just as there are good solutions and bad solutions there are Patterns and Anti-Patterns.
- Patterns are an Object Oriented concept. There is no reason not to think OO just because you are writing in Progress.

SOLVEPOINT

Why Anti-Patterns?

- It is often quicker and easier to solve problems initially with Anti-Patterns.
  - Time / Resource Constraints.
- Inherited Code.
- “If it ain’t broke don’t fix it”.
- We see these patterns over and over again in many Enterprises.

SOLVEPOINT

Anti-Patterns Can Cause

- Scalability issues.
- UI portability and Trading Partner Integration Problems.
- Buffer/Transaction scope bleeds.
- Difficulty in finding root causes of errors.
- Extended enhancement / maintenance coding times.
- Difficult to uncover or (maybe even worse) intermittent performance issues. These may extend to the entire Enterprise.

Types of Anti-Patterns

- Procedure Isolation / Coupling.
- Data Structure / Access.
- Block Oriented.
- We will discuss some very common anti-patterns that cause problems in enterprise settings.

SOLVEPOINT

Indirect Indirection

Caused by

- Nested or overuse of include files

Can Cause

- Scope bleed – Transaction/Lock (scary)
- Can be (very) Difficult to Maintain
- Can be (very) Difficult to Find Bugs
- Have a tendency to grow over time

SOLVEPOINT

Indirect Indirection

Root Causes

- Typically an organic problem, e.g. it is not planned it just happens over time.
- Solving part of the business problem instead of the whole problem due to time constraints or other reasons.
- Rush to coding before defining and understanding the entire business process.
- Lack of or miscommunication between business process resources and programming resources.
- Excessive cleverness / over engineering.

SOLVEPOINT

Indirect Indirection

Solution Pattern

- Use atomic well-defined procedures on an application server (or that are candidates for running on an app server as is) that solve or support the solution for the entire business problem.
- For existing code bases understanding the business need being addressed, fully, is imperative.
- Does this mean I should never use includes? No it does not, it means be very very careful about how and why you are using them.

SOLVEPOINT

Non-Isolated Sub Procedures

Caused By

- Sub procedures that free reference buffers or variables created or scoped outside of the sub procedure. This is really a variation of the Indirect Indirection pattern, it has many of the same symptoms.

Can Cause

- Buffer/Transaction scope bleed. Can cause limbo and deadlock.
- Unpredictable behavior, making it difficult to debug.
- Can cause data integrity issues if updates are predicated on a field changed somewhere else.

SOLVEPOINT

Non-Isolated Sub Procedures

Solution Pattern

- Sub Procedures should not reference, update, glance at, be aware of or otherwise try to use a variable or buffer not directly passed to them or that the sub procedure itself did not create or bring into scope.

- Note that app servers **enforce** this pattern by enforcing process isolation. This is a good thing.

SOLVEPOINT

(Not) Model-View-Controller

- What's Model-View-Controller (MVC)? MVC is the pattern that says keep your business logic (Model) and user-interface (View) separate.

Caused By

- Mixing Business Logic with User Interface code.

SOLVEPOINT

(Not) Model-View-Controller

Can Cause

- UI portability issues (e.g. give us a web / PDA etc interface).
- Trading Partner integration issues.
  - Web Services / XML Data Exchange.

Solution Pattern

- Ensure new code uses MVC.
  - Isolated atomic business logic on an app server (or a candidate to be put on an app server).

SOLVEPOINT

Functions in Blocks

Caused By

- Use of functions in a block or where statement.
  - Particularly on the left hand side of the assignment operator in a WHERE clause.

Can Cause

- Database Denial of Service to the Enterprise.
- Unintended whole table scans.
- Performance and scalability issues.
  - This can include sessions other than the session with the function reference if they are attached to the same database server process.

Database Denial of Service

- Because these are common patterns people may look at them and say ‘well, we see this all the time and it really hasn’t caused us a problem’.
- This example caused three days of serious enterprise wide database availability problems and took three days of four peoples time to the tune of forty+ resource-hours to locate…
- `IF CAN-FIND(FIRST account WHERE INTEGER(SomeFieldWithAlphasInIt + STRING(AcctNoAR)) = 127112) THEN .`
- This causes an error/retry loop in the database server process and results in thousands of entries per second being added to the .lg file of ‘Invalid Character in Numeric input A’.

Functions in Blocks

Performance

- But hey, do iCnt = 1 to num-entries(cChar) isn't a big deal, why should I care?
- It takes half the time to run an empty loop where num-entries is pre-determined than to run the same loop with num-entries being evaluated every time. (100 iterations at 150ms vs. 380ms in a simple test).
- Q. When is a 30ms operation too long? A. When you have to do it a million times.

SOLVEPOINT

Default Full Query Open

Caused By

- User Interfaces where the query backing a browse is open by default with no parameters.

Can Cause

- UI performance problems.
- Network congestion.
- Heavy database load / load spikes.
- Performance problems for other users attached to the same DB server process.

Default Full Query Open

Solution Pattern

- Do not default open the query. Other possible solutions include:
  - Attempt to determine a common set of ‘open’ parameters that make sense based on usage / function.
  - Remember user settings, e.g. of the last twenty times the user opened this query type they did so using xyz parameters.
  - Use application clues, e.g. user was viewing the account 034303 on the account view screen, so default to 034303 on the account transactions browse etc.

SOLVEPOINT

WAN DB Connections

Caused By

- Opening a client/server Progress session across a WAN connection e.g. a shared limited bandwidth leased line etc.

Can Cause

- Performance issues to include:
  - Deadlock
  - Slow user sessions
  - Long transactions
  - BI file growth
  - Etc.

SOLVEPOINT

WAN DB Connections

Solution Pattern

- Remote clients over a WAN connection should make Application Server calls to procedures residing in the LAN the database(s) live in.

- Application Servers will utilize an optimized compressed protocol for communicating via a slower WAN connection.

SOLVEPOINT

Default Block Error Handling

Caused By

- Coding DO, REPEAT, WHILE blocks with the default error handling properties.

Can Cause

- Un-handled un-logged session terminating errors.
  - You: “What error was on your screen?”
  - User: “Uh I clicked it off”
  - You: “But your not supposed to do that”
  - User: “I forgot”

SOLVEPOINT

Default Block Error Handling

Solution Pattern

- Use the on error, stop, quit undo retry option with an if retry block to log and handle the error in a friendlier way.

```
main:
DO ON ERROR UNDO main, RETRY main
   ON STOP  UNDO main, RETRY main
   ON QUIT  UNDO main, LEAVE main:

   IF RETRY THEN
   DO ON ERROR UNDO main, LEAVE main
      ON STOP  UNDO main, LEAVE main
      ON QUIT  UNDO main, LEAVE main:
      /* Log / Handle / RETURN or STOP etc */
   END.  /** RETRY **/
   /* Do work here */
END.  /** main **/
```

Default Block Error Handling

Solution Pattern cont.

- This block style can be nested to catch various types of errors you want to handle in different ways.

- Nesting in this way can be used to handle errors according to the type and severity. A caught error does depending on type may not have to end the session or operation.

SOLVEPOINT

Summary

Key Points

- Isolation of Atomic functions.
- Fine Granulation (but not too fine…).
- Does not rely on default error handling.
- Will scale under load.
- Address the current and future business need.
- Application Servers enforce many good coding behaviors, scalability, high availability, etc.

SOLVEPOINT

Root Causes

- Not considering performance / scalability
  - It's only for five users…today. What about in five years?
  - When is 30ms an operation too slow?
- Communication issues
  - Between business process stake holders and IT resources.
  - Miscommunication of specifications, rules, business needs.
- Time and resource issues
  - Fixing the error not the problem.
  - Make it work now and we'll fix it later (which means never).
  - Or, make it work this way for now, we know it's bad, when we do xyz we'll recode it.

SOLVEPOINT

Signs of Anti-Patterns

- The same code is continually being patched due to errors.
- Making small modifications or enhancements is a chore.
- A procedure does not scale well as load increases.
- Difficult or unable to integrate with other processes.

SOLVEPOINT

Question & Answer

Questions or Comments?

SOLVEPOINT

Thank You!

Thank you very much for your time and attention!

Solvepoint Corporation
882 South Matlack Street, Suite 110
West Chester, PA 19382

http://www.solvepoint.com

Building strong customer relationships through excellent service and delivery of advanced technology solutions.