Opening Remarks

• Small projects need to balance discipline & agility

• Customer wanted XP
  – Requirements are volatile and extremely risky
  – Requirements viewed as a learning exercise

• Developer wanted UP
  – How they do business

• Compromise reached

• Results to-date have been favorable
Setting the Stage – SSCA Project

• MDA/SMDC SBIR Phase II effort to protect applications software against:
  – Piracy
  – Tampering
  – Reverse engineering
• Contract aimed at showing that RCI’s software sneak circuit technology can scale without causing performance to degrade more than 20%
  – Measured by CPU, memory and cache utilization
What Are Sneak Circuits?

• Sneak circuits are tools used to obfuscate binaries
  – Make it hard for bad guys to reverse engineer the object code or executables
• Five categories of sneak circuits exist:
  – **Sneak data** – phantom data or incorrect or labeling of system inputs, controls or displays
  – **Sneak logic** – unintended logic flows
  – **Sneak timing** – unexpected tasks or conflicting sequencing of actions
  – **Sneak indicators** – ambiguous or false displays of operating conditions
  – **Sneak signatures** – false identification of compilers and tools used to generate/operate on binaries

Reference: NAVSO P-3634, Sneak Circuit Analysis, 1989
What Are the Common Threats?

- We have defined seven generic threats as part of the effort:
  - Data Recovery
  - Decompilation
  - Memory Overload
  - Processor Overload
  - Design Recovery
  - Cluster Analysis
  - Pattern Analysis
  - Slicing Analysis

- Proven and mature COTS tools and techniques exist to help the exploiters exploit often present vulnerabilities:
  - IDA Pro, Soft ICE, Win Hex, etc.
What Are the Major Benefits of Software Sneak Circuits?

- Obfuscate the code
  - Make it hard for others to develop logic maps, symbol tables and call-call trees
  - Develop deceptive timing and event sequences
  - Link to wrong messages and displays

- Provide forensic evidence
  - Route unauthorized users to honey pots

- Protect intellectual property and classified information
  - Make it hard for others to get to the family gems; the algorithms, classified data and “make” technology
Machine Code Hacking 101

- Easy to break into machine
- The easiest starting point is the FLASH ROM ("jam table" – op code interpreter)
  - Boot ROM contains hardware initialization routine, followed by instructions that load OS and often followed by OS itself
  - For example, on power up, Pentium starts running at reset vector location, 0xFFFF.FFFF0 (near top of memory)
    - Well documented by Intel and can easily be disassembled using hex editor or dissembler (using Hackman (freebie) or IDA Pro (more capable product))
  - Data at location is: 0xFFFF.FFF0 EBC6 8BFF 1800 D8FF FFFF 80C2 04B0 02EE
  - EBC6 is a jump instruction to location 0xFFFF.FFB8
  - The next chunk of code initializes the procedure’s GDT (Global Descriptor Table) and IDT (Interrupt Descriptor Table)
    - These tables are used to setup the processor’s memory management and interrupt handling schemes
- Like a ball of yarn, programs can be easily unwound if you know where the starting point is – it’s just a function of time & perseverance

Protection needed to preserve trusted execution of weapon system
The Sneak Circuit Project Team

Info Processing Ltd.
- Ada tool provider

Reifer Consultants, Inc.
PI – Donald Reifer

Absolute Software

CohesionForce, Inc.

Roles & Responsibilities
- Tool developer
- Working all aspects of development
- Support conduct of demonstration

Teledyne Solutions, Inc.

Roles & Responsibilities
- Transition agent for Phase III project
- Independent verifier that demo goals satisfied

(continued)
- Algorithms
- Requirements
- Commercialization
- Demonstration

Info Processing Ltd. (continued)
- Algorithms
- Requirements
- Commercialization
- Demonstration

Roles & Responsibilities
- Demo software developer
- Support conduct of demonstration
The Agile/XP Philosophy

**Manifesto**

- Individuals and Interactions over Processes and Tools
- Working software over Comprehensive documentation
- Customer collaboration over Contract negotiation
- Responding to change over Following a plan

Value Agility over Discipline
The Twelve Practices of XP

- Metaphor
- Release Planning
- Testing
- Pair Programming
- Refactoring
- Simple Design
- Collective Ownership
- Continuous Integration
- On-site Customer
- Small Releases
- 40-Hour Work Week
- Coding Standards

XP rewards demonstrable results; frequent demos of product software as it evolves
Characteristics of Agile Projects

- For the most part, agile methods projects could be characterized as:
  - **Short**: One year or less in duration (many shorter)
  - **Risky**
    - Requirements often are vague and volatile
    - Views refactoring as a normal part of the job
  - **Staffed with the high performers**
    - Motivated, experienced and committed troops
- Applications are somewhat unprecedented
  - Focus is on experimentation
- Projects characterized by high degree of required development flexibility
- Architecture is stable
- High degree of team cohesion
  - However, team may be geographically dispersed
- Some skepticism, but mostly enthusiastic support
**Unified Process Philosophy**

**UP Axioms**
- **Use Case & Risk Driven**
  - Know what the user requirements are
  - Consider risk items for early prototyping
  - Prioritize functionality
- **Architecture Centric**
  - Plan for architectural evolution
- **Iterative and Incremental**
  - Break problem into small chunks

**Best Practices**
- Develop software iteratively
- Manage requirements
- Use component based architecture
- Continuously verify software quality
- Control changes to software

**Plan-Driven Approach**
Unified Process Fundamentals

**Phased Development**
- Inception
- Elaboration
- Construction
- Transition

**Core Workflows**
- Business modeling
- Requirements
- Analysis & Design
- Implementation
- Test
- Deployment
- Configuration & Change Management
- Project Management
- Environment
Agile Meets Unified Process

**Agile/XP**
- Started with user stories/customer scenarios
- Initiated dialog via SOW with potential suppliers
  - Competitive procurement
  - Provide Huntsville test-bed
  - Asked for six week deliveries and demonstrations
  - Provided $$$ incentives
- Each supplier had a different approach
- Goal was risk reduction

**Unified Process**
- Appeared that use cases could be extracted from user stories
- Responded with proposal
  - Low staffing profile
  - Proposed 3 month iterations
  - Plan addressed high risk items first
  - Recommended low ceremony approach
- With tailoring, agility (XP) could be supported
Good Things

Agile/XP
• Product versus process focus yields results fast
• Risk reduction drives planning
• Demos provides focus
• Frequent releases provides feedback
• Continuous optimization via refactoring
• Shared processes for CM and documentation
• Dollar rewards for schedule achievement

Unified Process
• Iteration and prototyping yields results fast
• Use cases and risk reduction drives planning
• Analysis of big picture allows architecting for growth
• Frequent releases makes the problem smaller and allows risk to be reduced
• Founded on well known project management principles and systems
Systems and Procedures Implementation by the Project

- Project management
  - Cost, schedule and performance tracking
  - Risk management
- Monthly management reviews (in Huntsville)
  - Got sponsors involved
  - Action item tracking
- Requirements mgmt.
  - Baselines established and managed for threat, toolset and math algorithms
- Configuration mgmt.
  - Configuration identification
  - Change control
  - Problem reporting
  - Monthly CCB
- Quality evaluation
  - Metrics analysis
  - Independent testing
  - Independent verification
- Red teaming
  - Yet another assessment of how good the technology is
Delivery and Demo Schedule

• Cycle 1 (09/08/03 to 11/30/03)
  – Threat modeling
  – Requirements development
  – Prototype TMON GUI

• Cycle 2 (12/01/03 to 02/29/04)
  – Prototype metrics analysis
  – Prototype seed generator
  – Prototype DCON GUI

• Cycle 3 (03/01/04 to 5/31/04)
  – Finalize seed generator
  – Integrate metrics tools
  – Develop report generator
  – Develop DCON recorder
  – Prototype SPOT GUI

• Cycle 4 (06/01/04 to 08/31/04)
  – Develop seeding tools
  – Develop TGEN
  – Develop decoder
  – Integrate and test toolset

• Cycle 5 (09/01/04 to 11/30/04)
  – Evaluate effectiveness of SSCA protection
  – Get ready for demo
  – Develop draft User Guide

• Cycle 6 (12/01/04 to 2/28/05)
  – Conduct demo
  – Deliver software
  – Start commercialization
What Did We Develop?

- Contract deliverables
  - Quarterly progress reports
- Presentations
  - Reviews and pitches
- Procedures
  - CM, QA, testing, etc.
- Logs
  - Action item, problem report and product status
- Reports
  - User scenarios
  - Threat definition
  - Strategy vector definition
  - Seeding algorithms and decision logic specification
  - Meeting minutes, etc.
  - Background materials
- Manuals
  - Threat generator guide
  - Toolset user’s guide
- Web site for collaboration
  - Staff worked at home
  - Subcontractor in Huntsville
  - Sponsor in Washington, DC
- Software
  - Tools
    - SCADS – 30 KSLOC of code written in C and Java
    - TGEN – 10 KSLOC of code written in C and Java
    - TMON – 10 KSLOC of code written in Ada
  - From 8/03 to 2/05, twenty-one software deliveries were made and managed
  - About a delivery a month
The Software Toolset

- TMS (Temperature Monitor System)
  - Ada application from Absolute Software
  - GUI built by CFI (TMON)
- TGEN (Threat Generator)
- SCADS (Sneak Circuit Analysis Demonstration Support)
  - SPOT (Software Prototype Obfuscation Toolset)
  - DCON (Demonstration Controller)
  - PMON (Performance Monitor)
What Actually Happened?

• We got behind
  – Too much to do
  – Not enough staff to do it

• How we caught up
  – RCI developed the threat generator in Cycle 5 to offload CFI
  – Integration and test done by combined team in Huntsville (more than 40 hour weeks)
  – Documentation pushed until after demo in Cycle 6
  – Some non-essential capabilities were deferred

• Our process saved our bacon
  – We knew we were behind and took corrective actions
  – We had control over the product and its quality
  – We managed risk and dealt with it head-on

• What we delivered
  – A successful technology demonstration
  – A toolset and associated documentation
  – Follow-on business
## Not So Good Things

### Agile/XP
- Lack of focus on the overall architecture
  - Metaphor doesn’t provide enough context for design
  - Use cases helped here
- Team geographically dispersed in 3 locales
  - Having customer on-site not feasible
- Many distractions
  - Briefings to interested parties/weekly exercises
- At first, too many formal versus working meetings

### Unified Process
- Rigid plans can lead to cumbersome project evolution
  - Need to iterate and continuously refactor
- Maintenance of unnecessary artifacts drains valuable time and schedule
  - Product versus document focus needs to be balanced
- Requires training of all team members on agreed to process & techniques
## Compromises

### Agile/XP
- Three month versus six week delivery cycles
  - With interim releases in between every few weeks
- Specs supplement user stories/customer demo scenarios
- Peer reviews vs. pairs because of co-location issues
- Monthly working session versus on-site customer reviews/interference
- Shared processes that embodied compromises

### Unified Process
- Less project architecture description
- Less focus on formal design documentation
- Project pace requires frequent strong pushes to meet demo requirements
- Plans change more frequently due to learning and refactoring
- Requirements evolve throughout project as learning takes place
- More flexibility, less ceremony
Lessons Learned

**Agile/XP**
- XP and UP when used together can still be agile
- Hybrid approach takes advantage of strengths of both approaches
  - Use cases help a lot
  - So does frequent delivery and demo
- Key is to be flexible and do what’s right, not what’s popular
- Emphasis on product and frequent releases pays off when risky

**Unified Process**
- Building a process using best practices is better than attempting to tailor out what is not needed
- Architecture stability is critical when performing evolutionary research
- Although UP supports large projects, it can be scaled down successfully
- When the going gets tough, resist the temptation to say the heck with the process
Would You Use a XP/UP Methodology Again?

- **Would You Use?**
  - Yes, we would do it again

- **Who is You?**
  - RCI – prime contractor
  - Test bed in Huntsville at customer site

- **What Products?**
  - Very good for projects where requirements are evolving/volatile
  - Development of tools that automate protection technology
  - Other products where the method is applicable

- **Under What Conditions?**
  - Small to medium sized technology demo project
    - Less than 10 people, 50 KSLOC
  - Requirements are volatile
  - Architecture is fluid
  - Focus is supporting reducing risk and conduct of a successful demonstration

- **Why?**
  - Need to iterate based on experimental results
  - Need to show not tell the customer our progress
Some Recommendations

- Clearly understand what is meant by agile/UP methods
  - Variants/invariants
- Fit methods properly
  - Use lessons learned
- Focus on capturing metrics/“hard” data
  - Use to update plans
- Focus on optimization via refactoring

- Introduce methods slowly and carefully
  - Address learning curve
  - Provide startup guides and “how to” checklists
- Focus on architecture
- Make sure methods are compatible with your process
- Do what makes good business sense
Final Remarks

• “Technology travels with people. You can’t just throw it over the wall and, because it is a good idea, expect people to pick it up and run with it.”
  
  Chuck Geschke, Co-Founder of Adobe Systems

• “If you don’t know where you are, a map won’t help.”
  
  Watts Humphrey, SEI

• “You’ve got to be very careful if you don’t know where you’re going, because you might not get there.”
  
  Yogi Berra, New York Yankees

• “It takes courage to do what is right.”
  
  Donald Reifer, Presenter