Software Reuse in BIOS using Program Families and Software Product Lines

Lee Rosenbaum
Software Engineer
Intel
June 24, 2010

Last Updated: Oct 23, 2009

Legal Notices

This presentation is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino logo, Core Inside, Dialogic, FlashFile, i960, InstantIP, Intel, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, IPLink, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skoool, Sound Mark, The Journey Inside, VTune, Xeon, and Xeon Inside are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Last Updated: Aug 28, 2006

Agenda

- *Executive Summary*
- *Difficulties of Software Reuse*
- *Program Families and Software Product Lines*
- *Reuse in the UEFI/PIWG BIOS Architecture*
- *Conclusions and Opportunities*

Executive Summary

- *Software Reuse needs to be planned*
- *Techniques Exist: Program Families and Software Product Lines*
- *In the BIOS domain, the UEFI/PIWG Architecture is one example of a product line*
- *Other examples cited in backup section*

The Challenge:

- Maximize software reuse across a set of related products.

The Problem:

- Complex SW is essentially difficult
- Processes and approaches often discourage reuse
  - Craftsman approach
  - Single product focus

Focus on reuse at the architectural level

Planning for Reuse

Who does reuse affect?

- Validation teams
- Product teams
- Software teams
- Customers

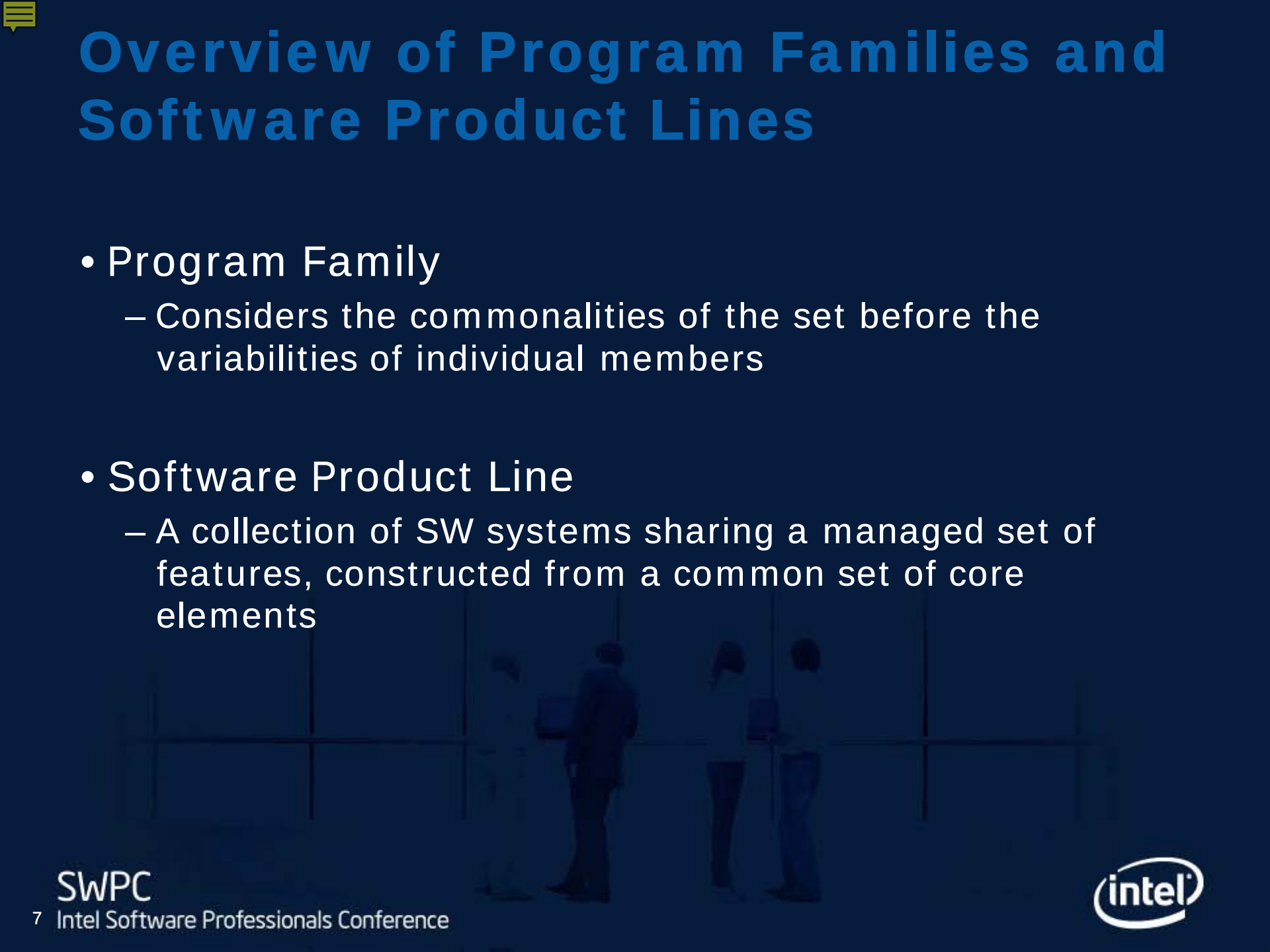
Why is it important?

- Reduces development time and improves quality by allowing working code to be reused

Reuse across products can be increased with planning

Overview of Program Families and Software Product Lines

- Program Family
  - Considers the commonalities of the set before the variabilities of individual members
- Software Product Line
  - A collection of SW systems sharing a managed set of features, constructed from a common set of core elements



Product Line Development Process

- Domain Engineering Phase
  - Investment Phase
  - Define the family and level of production
- Develop Application Engineering Environment
  - Defines how each PL Instance will be developed
- Application Engineering
  - Payback phase
  - Produce each family instance

Lightweight adoption - requires more expertise & development for each instance than a full PL process

Reuse in *the UEFI/PIWG Architecture*

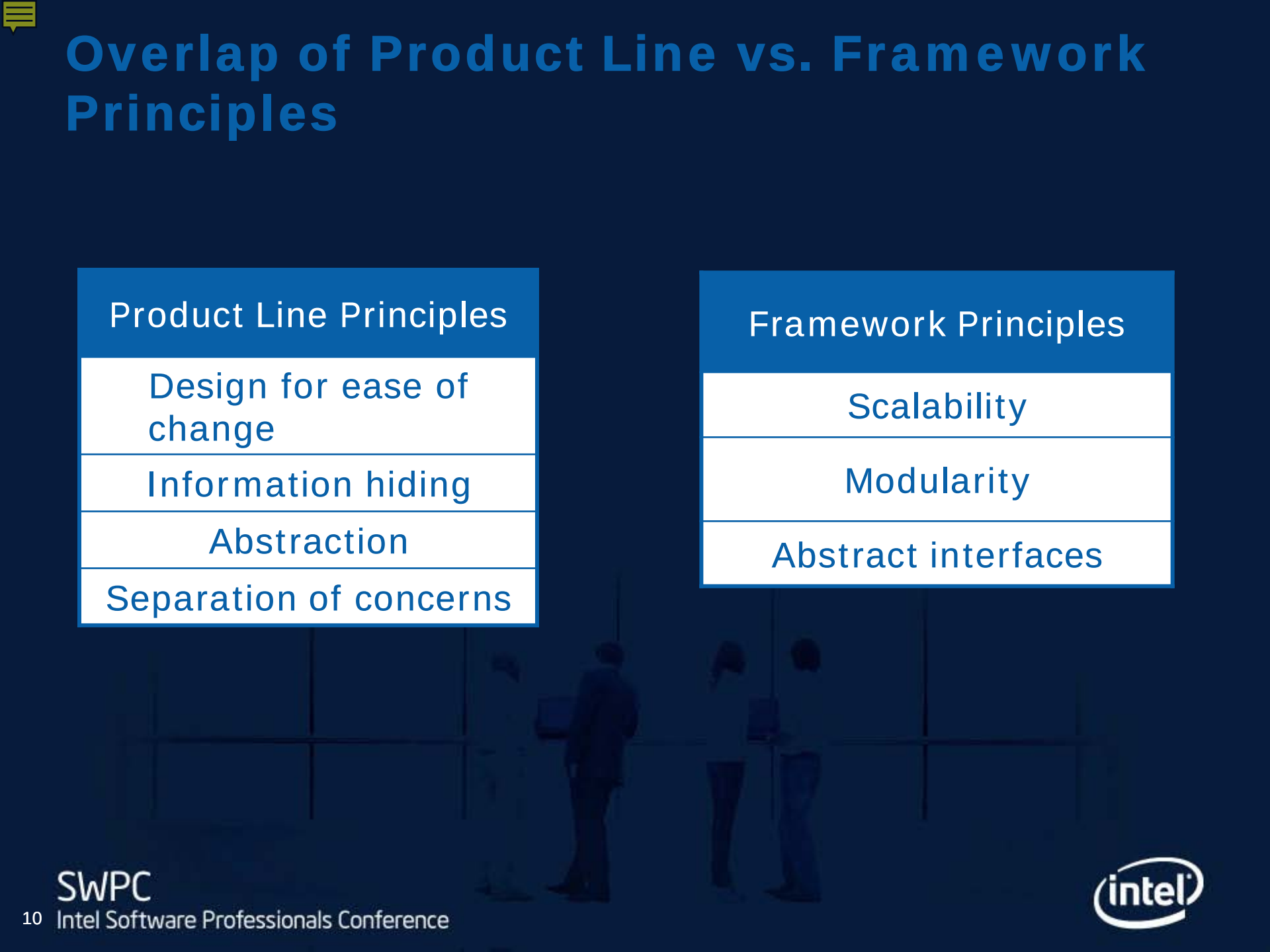
- Well defined set of extensible interfaces
- Common set of phase appropriate services
- Decouples:
  - SW abstractions from micro-architectures, HW interfaces, industry standards & platform topologies
  - OS from BIOS via standardized abstract interface.
- Modules collected into platforms using build description files
- Platform configuration data to specify variabilities

Open sourced / standardized BIOS infrastructure

Overlap of Product Line vs. Framework Principles

Product Line Principles
Design for ease of change
Information hiding
Abstraction
Separation of concerns

Framework Principles
Scalability
Modularity
Abstract interfaces



Adoption & Successes 1

- UEFI.org
  - Promoters: AMD, Intel, Apple, Dell, HP, IBM, Lenovo, AMI, Phoenix, Insyde, Microsoft
  - total member companies: 120+
- During 2009, > 50% of systems shipped will be UEFI compliant
- BIOS as distinguishing product feature
  - Apple boot camp capability allowing MAC's to boot Windows

Adoption & Successes 2

- One IBV reported:
  - "*100% common non-Silicon code across IA32 & X64 platforms [Atom to Xeon*]"
  - Reduced training and time to market
- 70% shared code
  - between Itanium [server] & X64 [workstation] reference platforms
  - with no common Si
- 85% shared code
  - if only difference is processor architecture

Summary

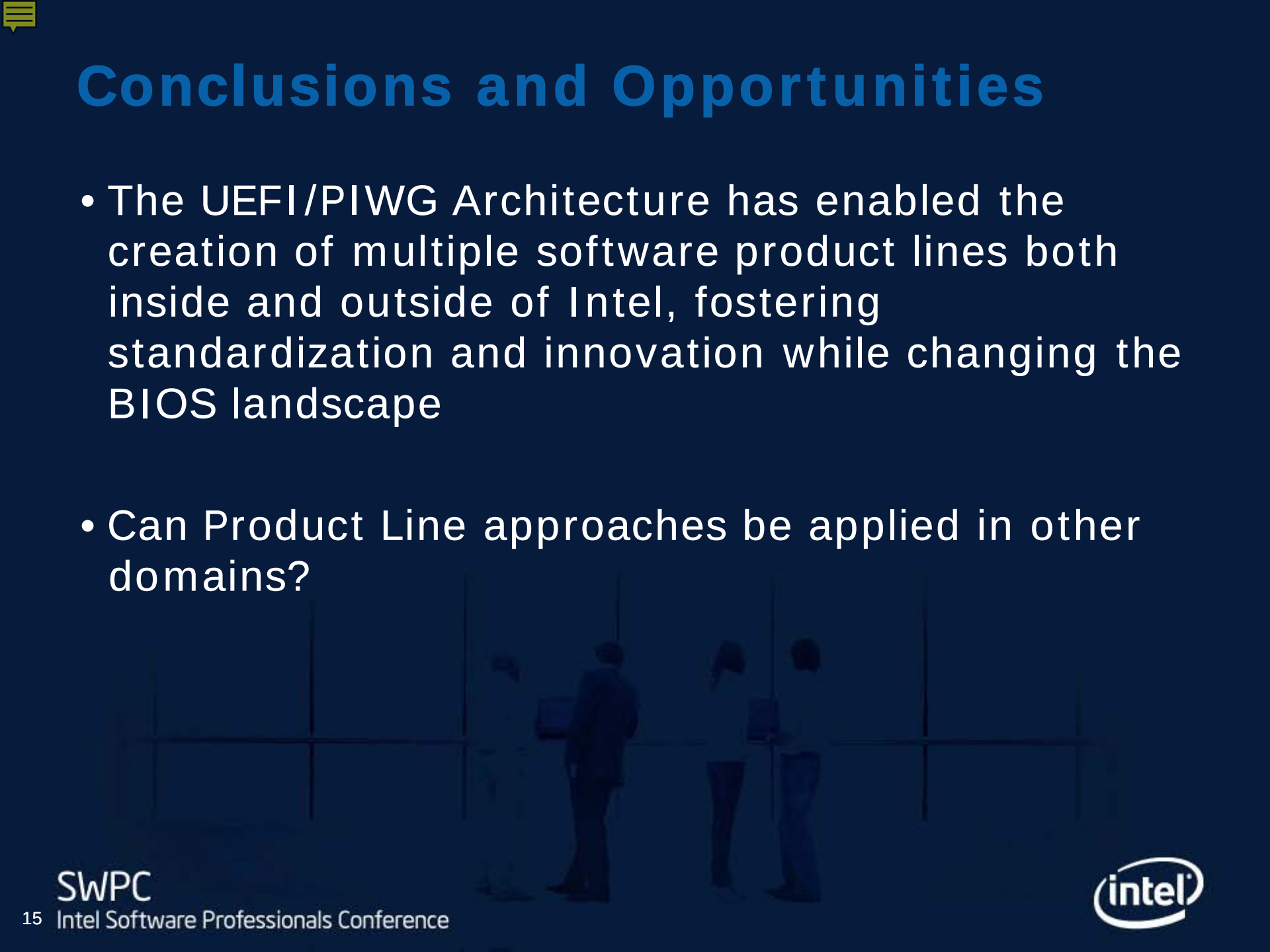
Lessons Learned

- Organization & process changes are often needed
- Expect payback after 2 or 3 instances
- Need to deal with resistance to change:
  - New concepts, languages, tools, techniques
  - Code generation, hidden Makefiles, Wizards …
- Minimize support of multiple generations and versions
- Framework transition complicated by legacy BIOS issues and industry standards process
- Creates new opportunities

Conclusions and Opportunities

- The UEFI/PIWG Architecture has enabled the creation of multiple software product lines both inside and outside of Intel, fostering standardization and innovation while changing the BIOS landscape

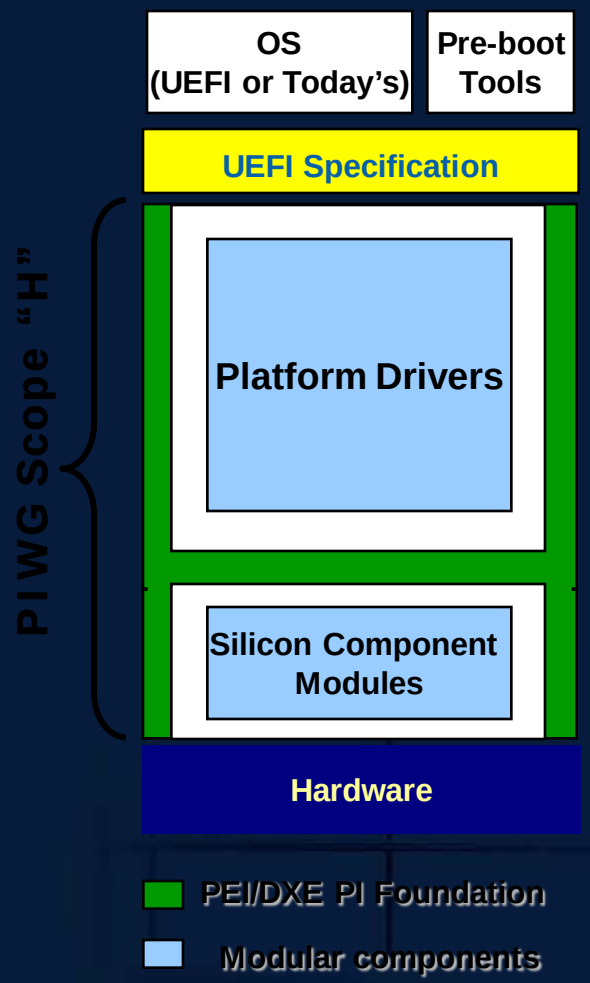
- Can Product Line approaches be applied in other domains?



Software Reuse in BIOS using Software Product Lines

Backup

Intel's UEFI Framework Architecture



PIWG – defines the platform initialization infrastructure beneath the UEFI spec.

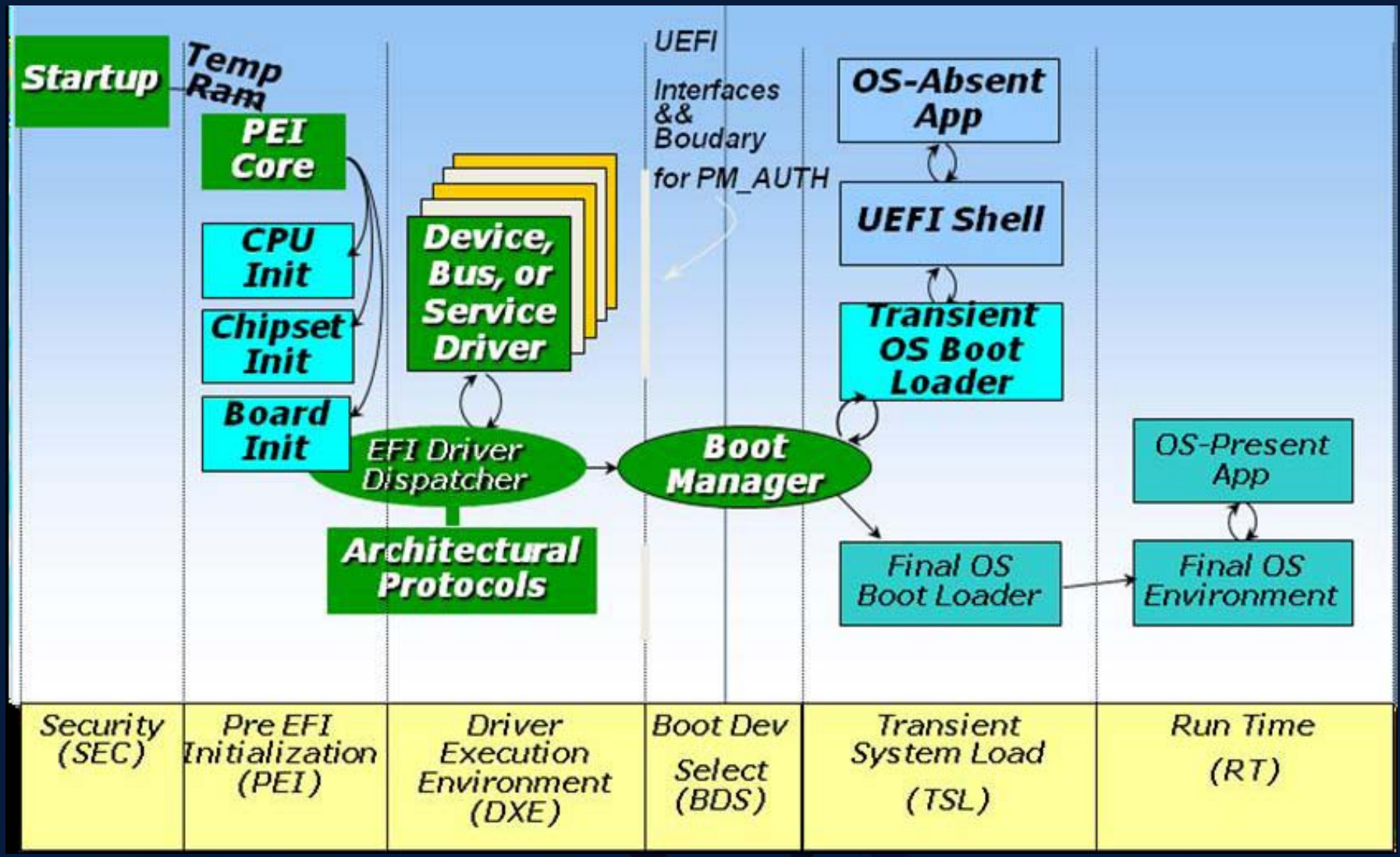
➔DXE Layer

Driver E**x**ecution **E**nvironment

➔PEI Layer

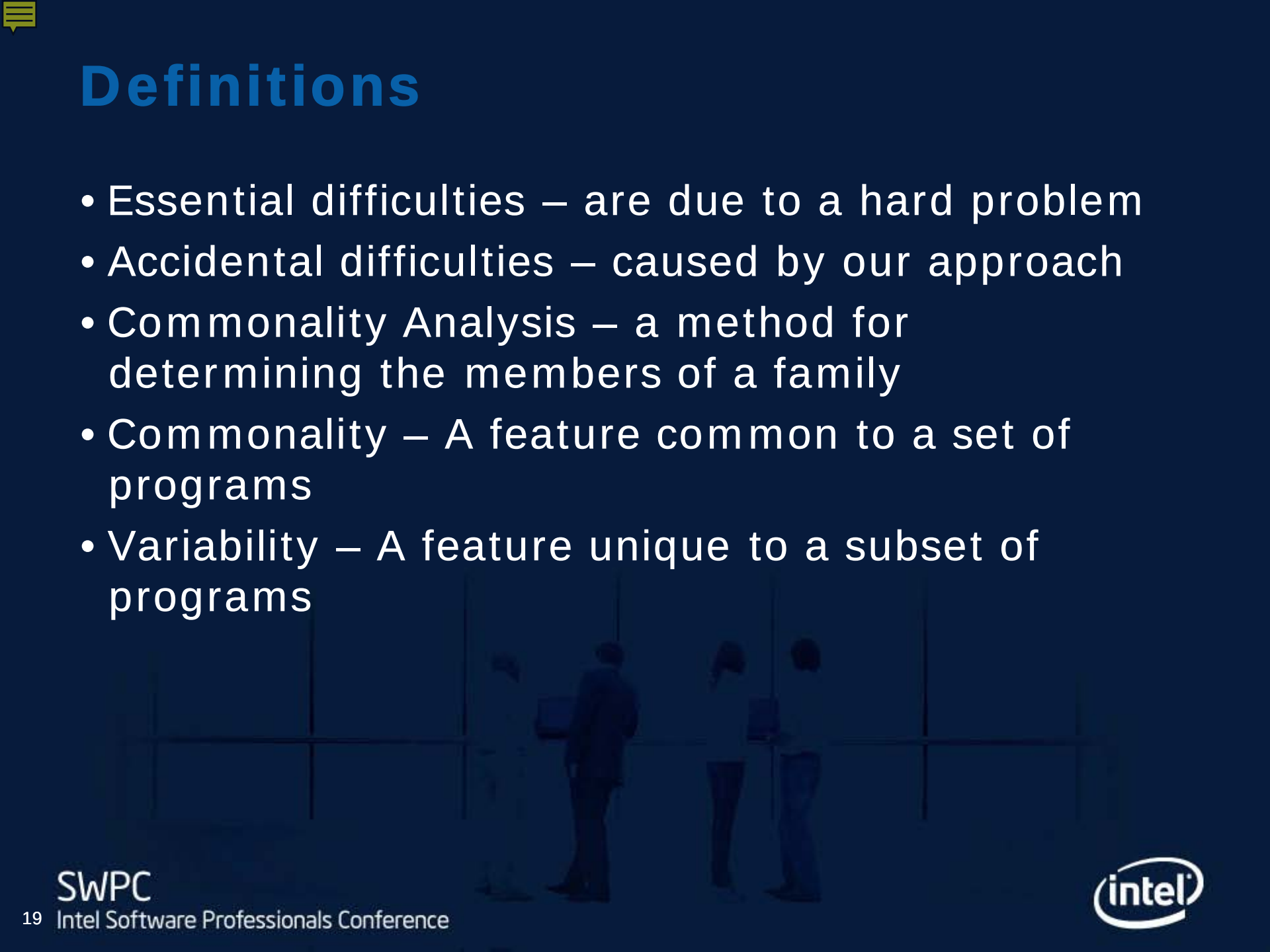
Pre-**E**FI **I**nitialization

Boot Flow


Startup
Temp Ram
PEI Core
CPU Init
Chipset Init
Board Init
Device, Bus, or Service Driver
EFI Driver Dispatcher
Architectural Protocols
UEFI
Interfaces && Boudary for PM_AUTH
Boot Manager
OS-Absent App
UEFI Shell
Transient OS Boot Loader
Final OS Boot Loader
OS-Present App
Final OS Environment
Security (SEC)
Pre EFI Initialization (PEI)
Driver Execution Environment (DXE)
Boot Dev Select (BDS)
Transient System Load (TSL)
Run Time (RT)

Definitions

- Essential difficulties – are due to a hard problem
- Accidental difficulties – caused by our approach
- Commonality Analysis – a method for determining the members of a family
- Commonality – A feature common to a set of programs
- Variability – A feature unique to a subset of programs

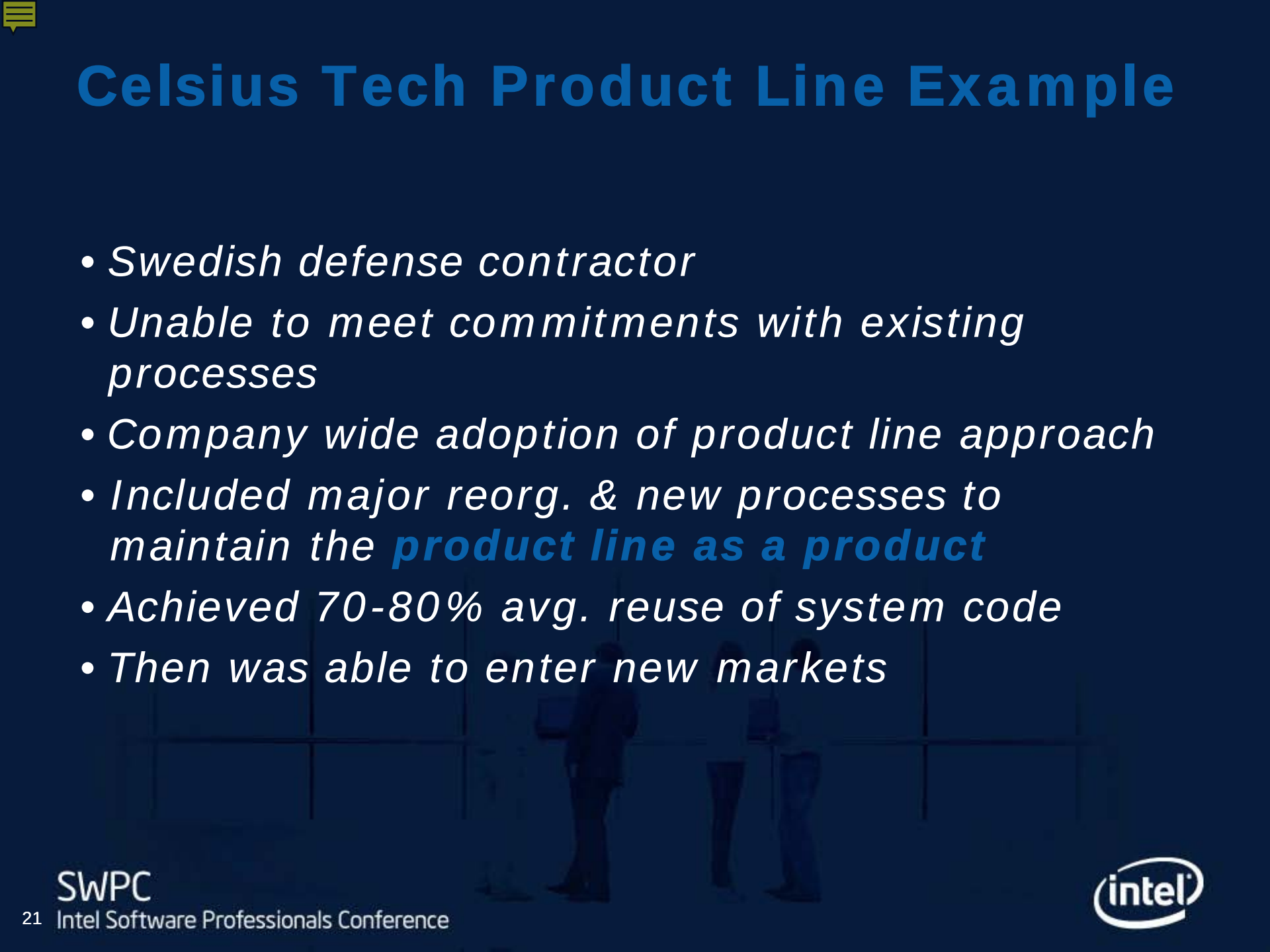


Definitions

- Program Family – A set of programs structured based on their commonalities and variabilities
- SW Product Line – A set of programs constructed from a common set of core elements
- Domain - a group of products with a well known & generally accepted set of features.

Celsius Tech Product Line Example

- *Swedish defense contractor*
- *Unable to meet commitments with existing processes*
- *Company wide adoption of product line approach*
- *Included major reorg. & new processes to maintain the **product line as a product***
- *Achieved 70-80% avg. reuse of system code*
- *Then was able to enter new markets*



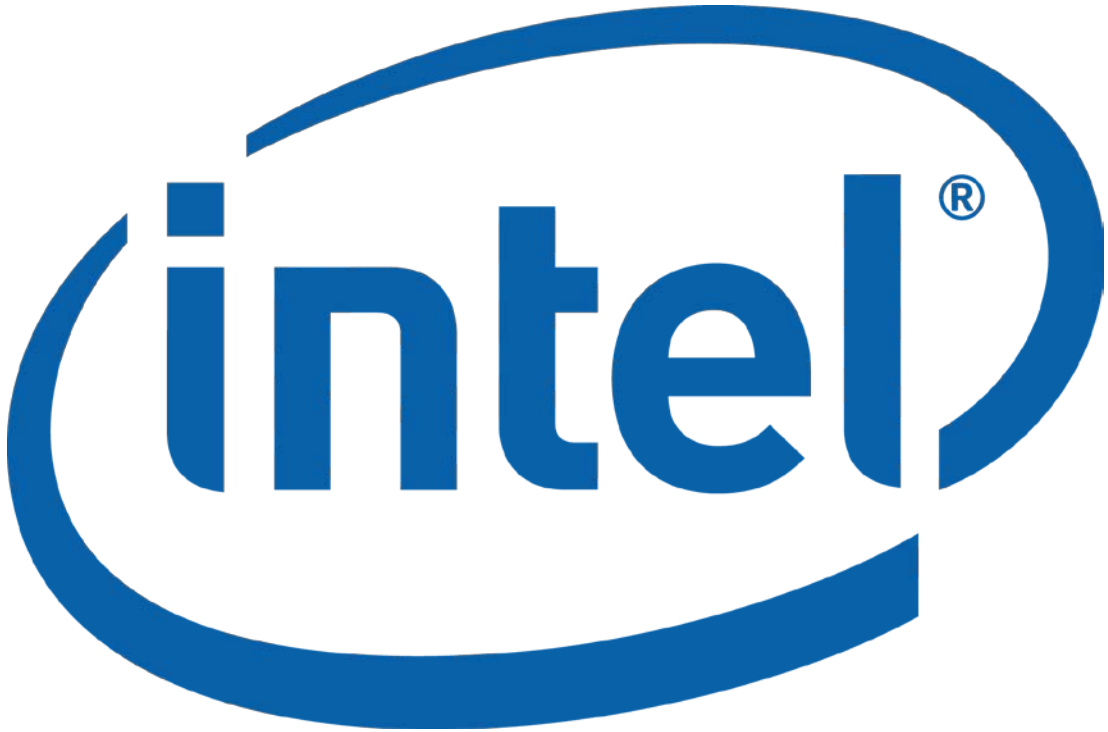
Additional Information

- *Software Product Line Engineering, A Family-Based Software Development Process*, Weiss & Lai
- *On the Design and Development of Program Families*, *Software Fundamentals Collected Papers of David Parnas*
- www.softwareproductlines.com
- http://www.pdx.edu/omse/courses#551

- Contact Information: lee.g.rosenbaum@intel.com

Acknowledgements

- Co-author: Vincent Zimmer
- Reviewer: Lawrence Meadows


intel