intelligence everywhere™

Software Quality and Reliability

Terry Heng
Senior VP & General Manager
Motorola Global Software Group

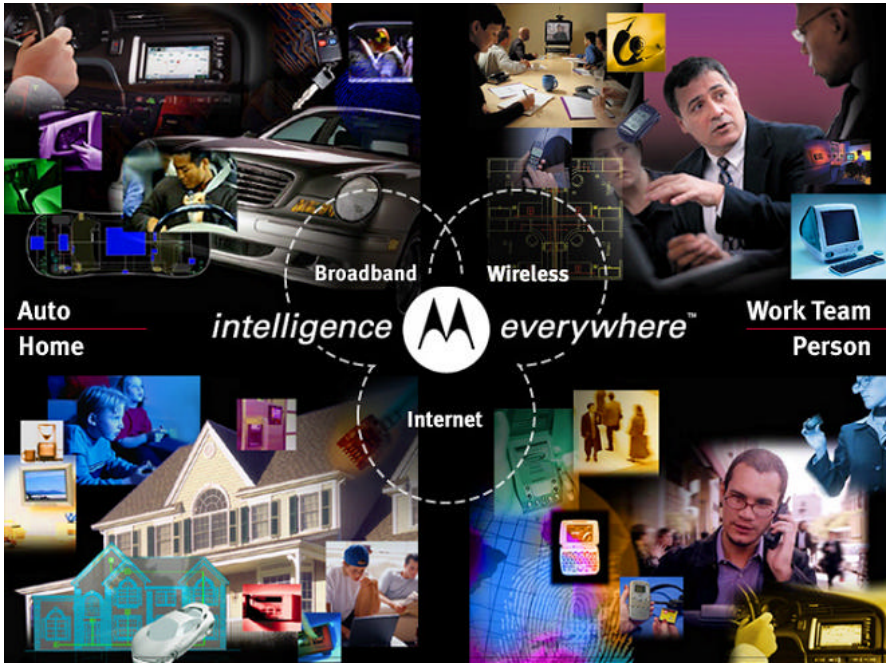
28 November 2001

MOTOROLA

intelligence everywhere™

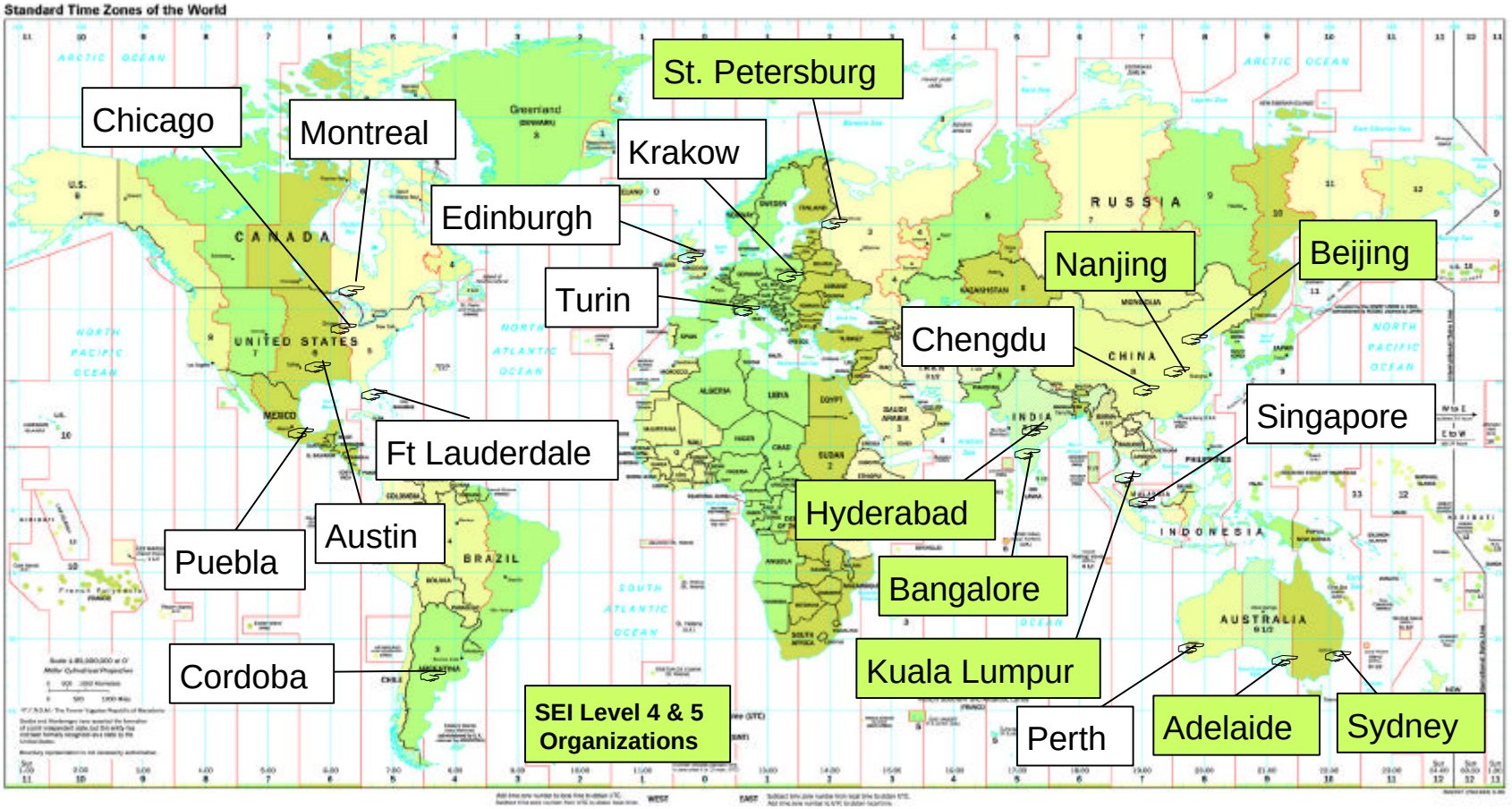
GSG Vision

Be the premier provider of innovative software products and services to Motorola's businesses and customers worldwide.



intelligence everywhere™

GSG Locations World-Wide (over 65% SEI Level 4/5)



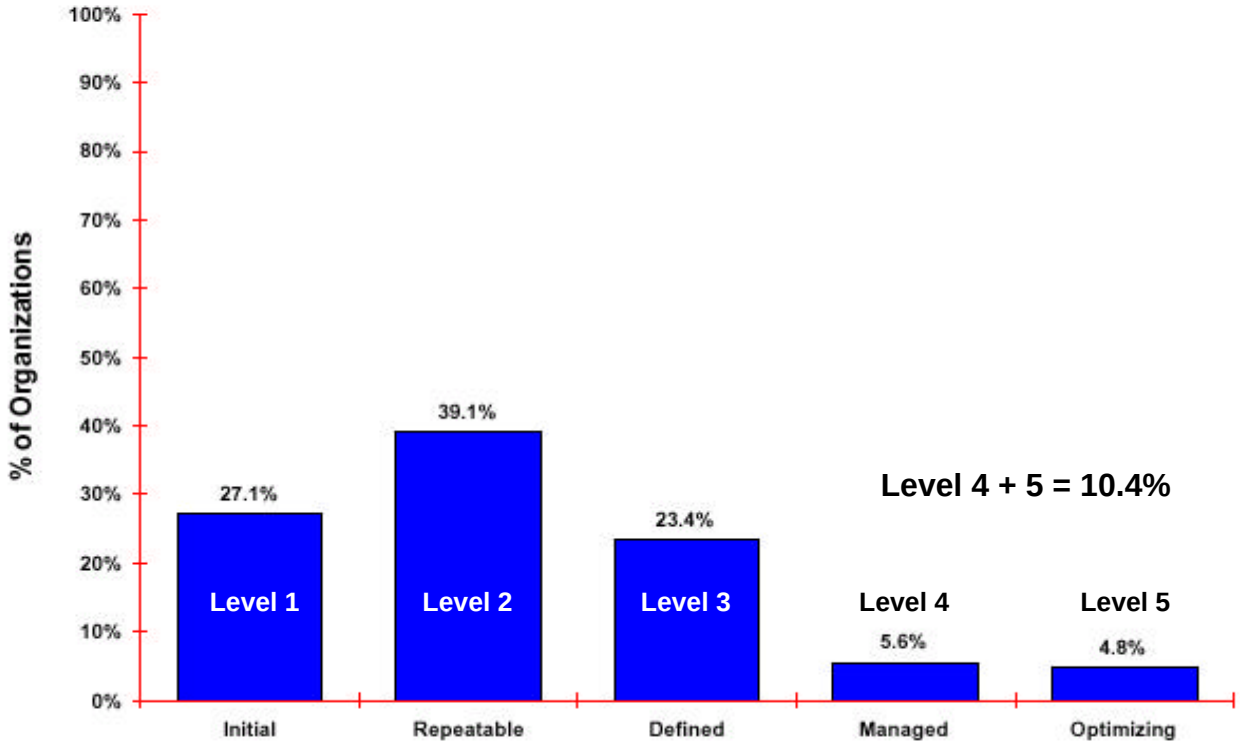
intelligence everywhere™

Carnegie Mellon University
Software Engineering Institute

Summary Industry SEI Data

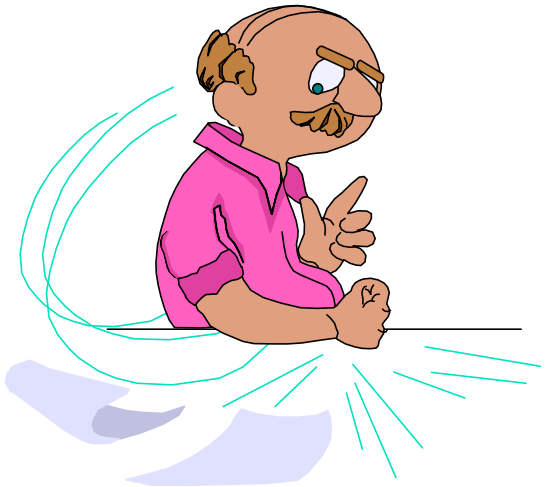
Organization Maturity Profile

August 2001



Based on most recent assessment, since 1997, of 1018 organizations. For a perspective, please see page 18.

Why is Software Reliability Engineering (SRE) Important?



- Defective software cost industry worldwide $175 billion in year 2000.
- Loss of a single network cell site costs $18K per minute of downtime.
- More than 110 million computers are on-line and connected via the Internet today, and are prone to virus attacks and defects.
- >90% of institutions reported some insider abuse of network access in year 2000.

- Loss of competitive position and market share
- Unsatisfactory return on software investment
- High cost of defects discovered late in development or in use
- Poor-quality products and slow response to our customers' needs (increased functionality)
- Missed schedules
- Low availability
- Lack of security / fault protection

intelligence everywhere™

High Availability Software in GSG

System Availability **99.999%** of the Time

- The **5NINES Initiative** in Motorola is aimed at increasing the availability of Motorola's products and systems to 99.999%
  - to satisfy our customers
  - to achieve business success in today's competitive wireless and broadband Telecom marketplace
- The **GSG High Reliability and Availability Technology Center** supports the 5NINES Initiative by
  - partnering with Motorola sectors to transfer promising technologies into their products and processes
  - serving as a focus for practical education
  - pushing the envelope

Key Industry Challenges to Improve Software Reliability

- **Reduce or eliminate defects from software – improve software quality**
- **Design software for reliability, fault tolerance, and rapid fault recovery**
- **Maximize the use of proven SRE models and techniques in industry**
- **Allocate V&V (Verification & Validation) resources for maximum payoff**
- **Invest in Academia to improve payoff from SRE models and theory**

Key Industry Challenges to Improve Software Reliability

- **Reduce or eliminate defects from software – improve software quality**
  - Monitor and constantly improve software cost of quality
  - Conduct root cause analyses, perform defect prevention
  - Adopt approaches like the SEI model to drive software capability
- **Design software for reliability, fault tolerance, and rapid fault recovery**
  - Allocate software reliability at the architectural design stage
  - Perform software Failure Mode and Effects Analysis (FMEA)
  - Develop fault management systems and self-correcting software designs
- **Maximize the use of proven SRE models and techniques in industry**
  - Create internal SRE experts to work with projects (Blackbelts)
  - Task system engineers and architects to analyze operational profiles
  - Adopt and use existing software reliability tools/techniques (CASRE, Ultrasan, …)
- **Allocate V&V (Verification & Validation) resources for maximum payoff**
  - Conduct profile-oriented reviews and testing
  - Identify and eliminate “hot spots” in software
  - Set thresholds for re-entering inspection, design, or analysis
- **Invest in Academia to improve payoff from SRE models and theory**

Linking the SEI Model with Software Reliability Engineering (SRE)

SEI	SRE
• Level 2 – Software Rqt's/Project Management, CM, and Quality	• Use Defect Data to Build Initial SRE Models, Assess Field Quality
• Level 3 – Software Development and Design Engineering, V&V	• Design for Reliability, Predict Quality via Use Cases & Operational Profiles
• Level 4 – Software Development Statistical Process Control*	• Enhance SRE Models, Use SRE to Determine Optimal Testing Regime
• Level 5 – Continuous Software Improvement, Prevention of Defects, Technology and Process Change Management*	• Adopt Advanced Methods to Create High-Availability Systems (5 NINES) and Drive Elimination of Operational Faults (FMEA)
• Future – Advanced SEI Process Models (PSP/TSP, CMMI, …)	• Develop Component/Product/System SRE Models, Fault-Tolerant Methods

*** Note: Good data is often hard to achieve until an organization reaches Level 4.**

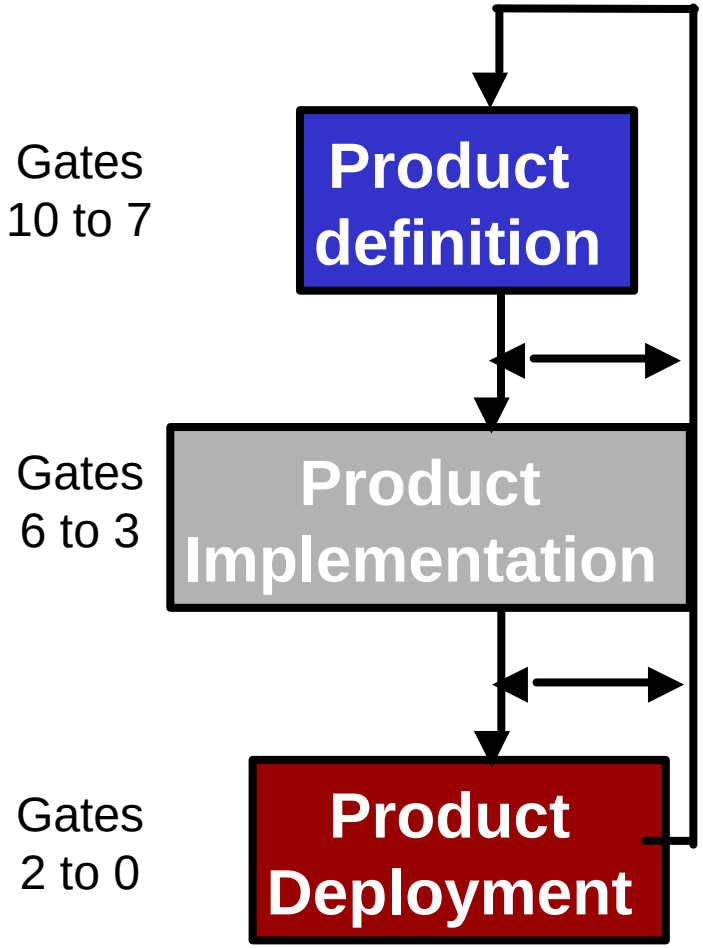
Results to Date from GSG Process Improvement

Attribute	1993	1995	2000	Industry Average
Delivered Quality Level	5.1?	5.7?	5.9?	4.3?
Cost of Poor Quality	35%	17%	6-8%	40%
Relative Productivity	1.5X	2.2X	2.6-6X	1X
Cycle Time Improvement	2.75X	4.4X	6X+	N/A

Plus 90% on-time delivery of 300+ projects over the past 6 years.

intelligence everywhere™

Iterative intervention to improve reliability and availability



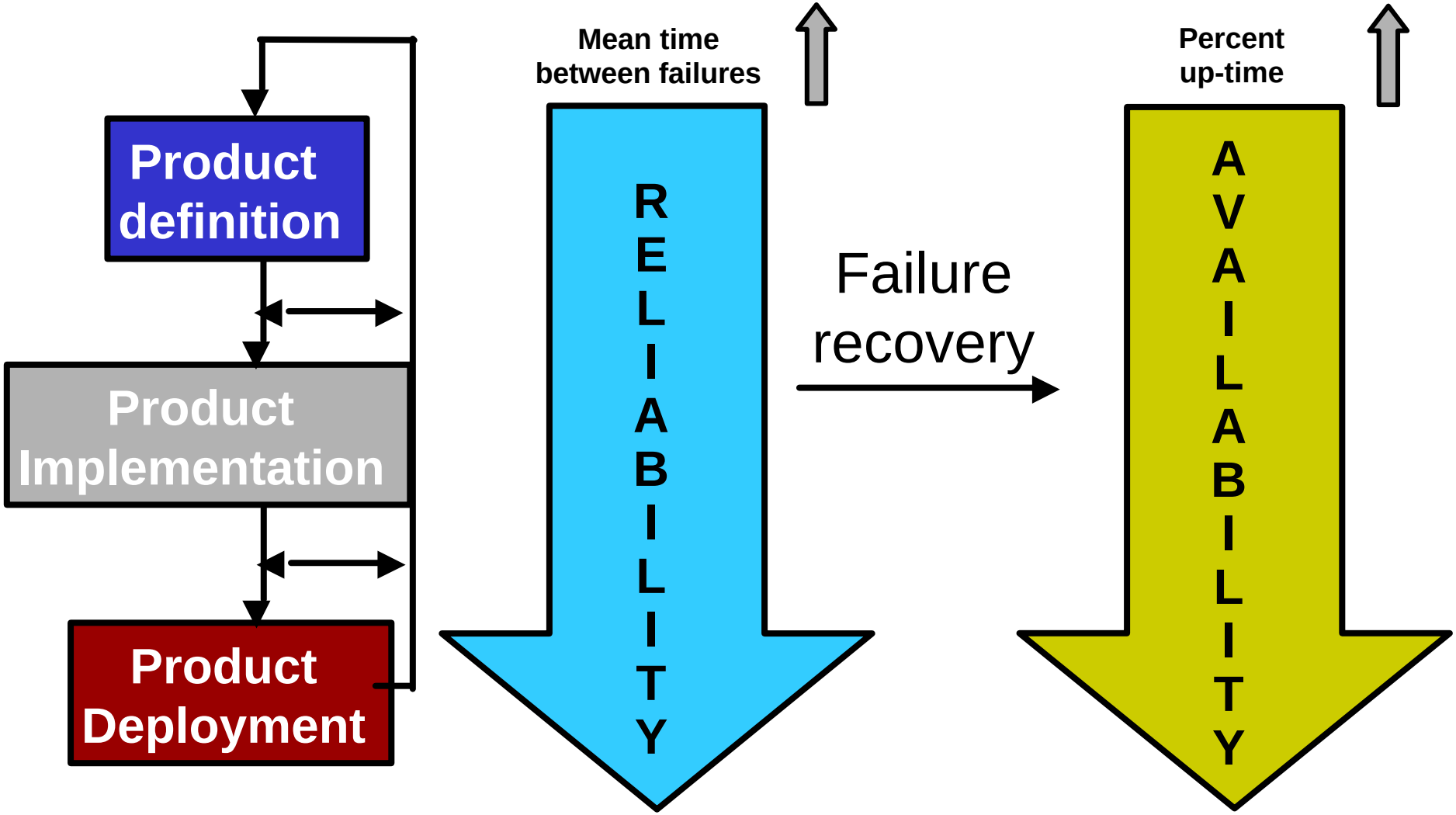
- System architecture improvements
- System architecture assessment

- SW and HW fault tolerance methods
- Lab assessment through fault insertion

- Improve field installations
- Field availability measurement & analysis

intelligence everywhere™

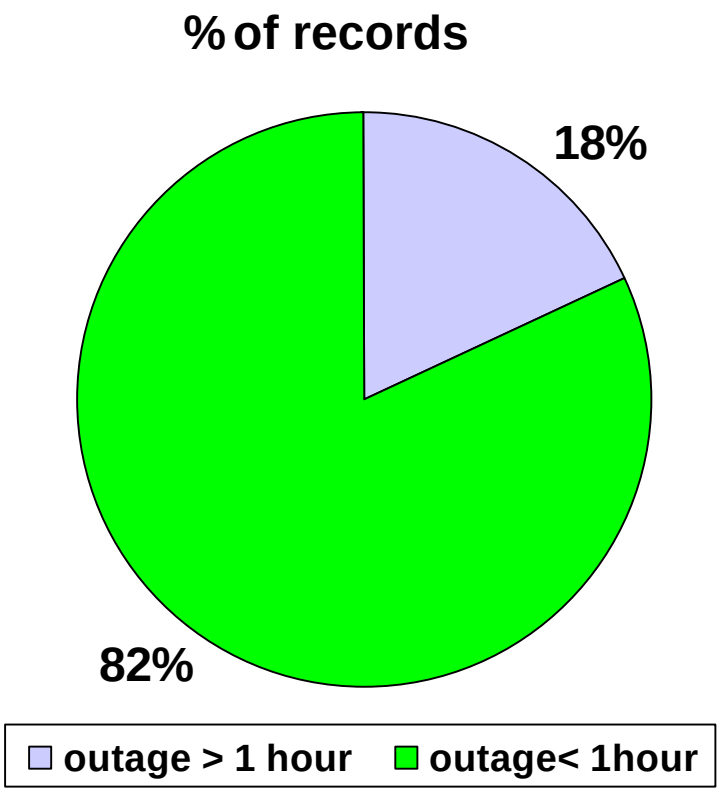
Two related essential tracks

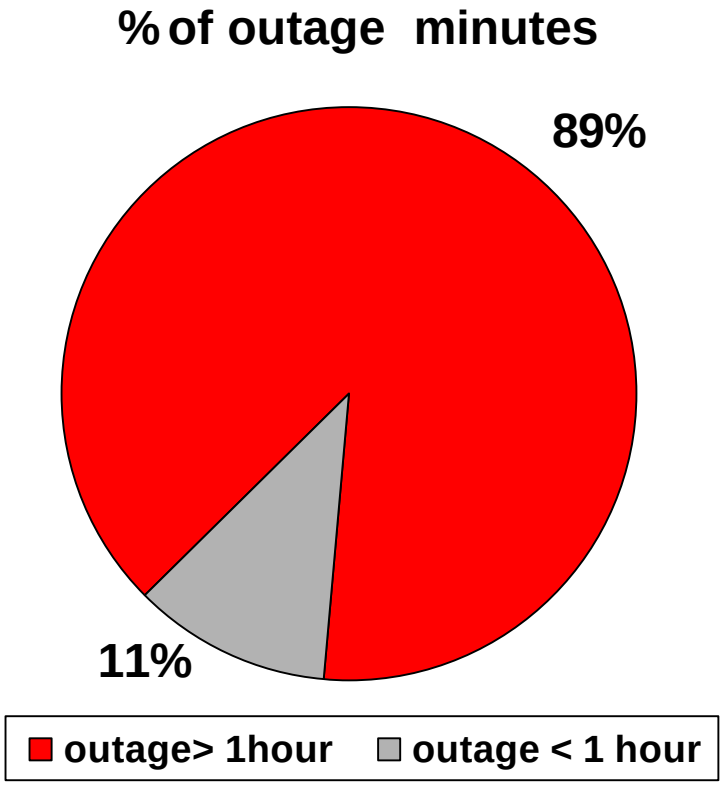


intelligence everywhere™

Example GSG-Motorola Sector outage data analysis

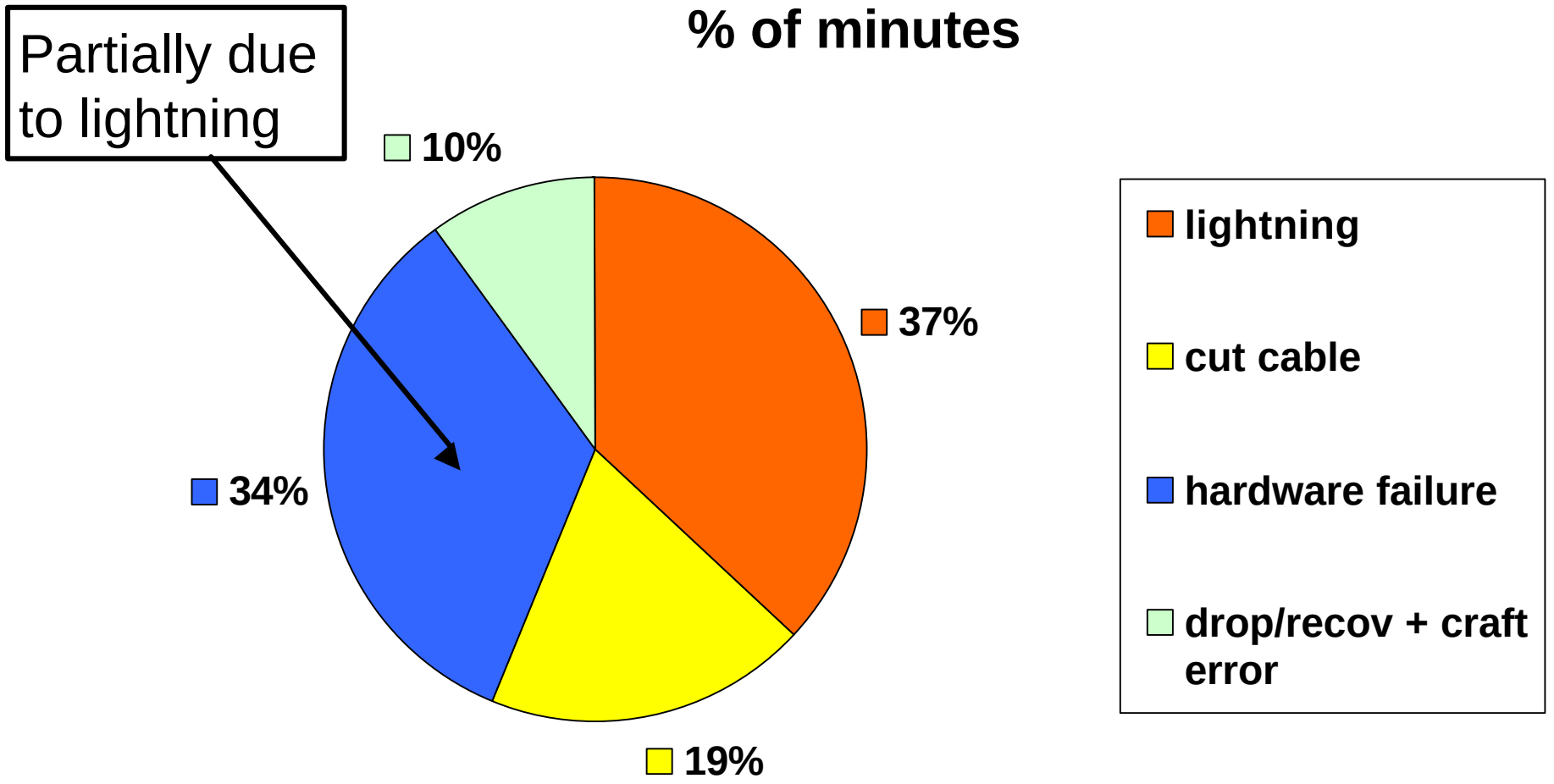
Jan 1999 - March 2000
Current availability 99.85%





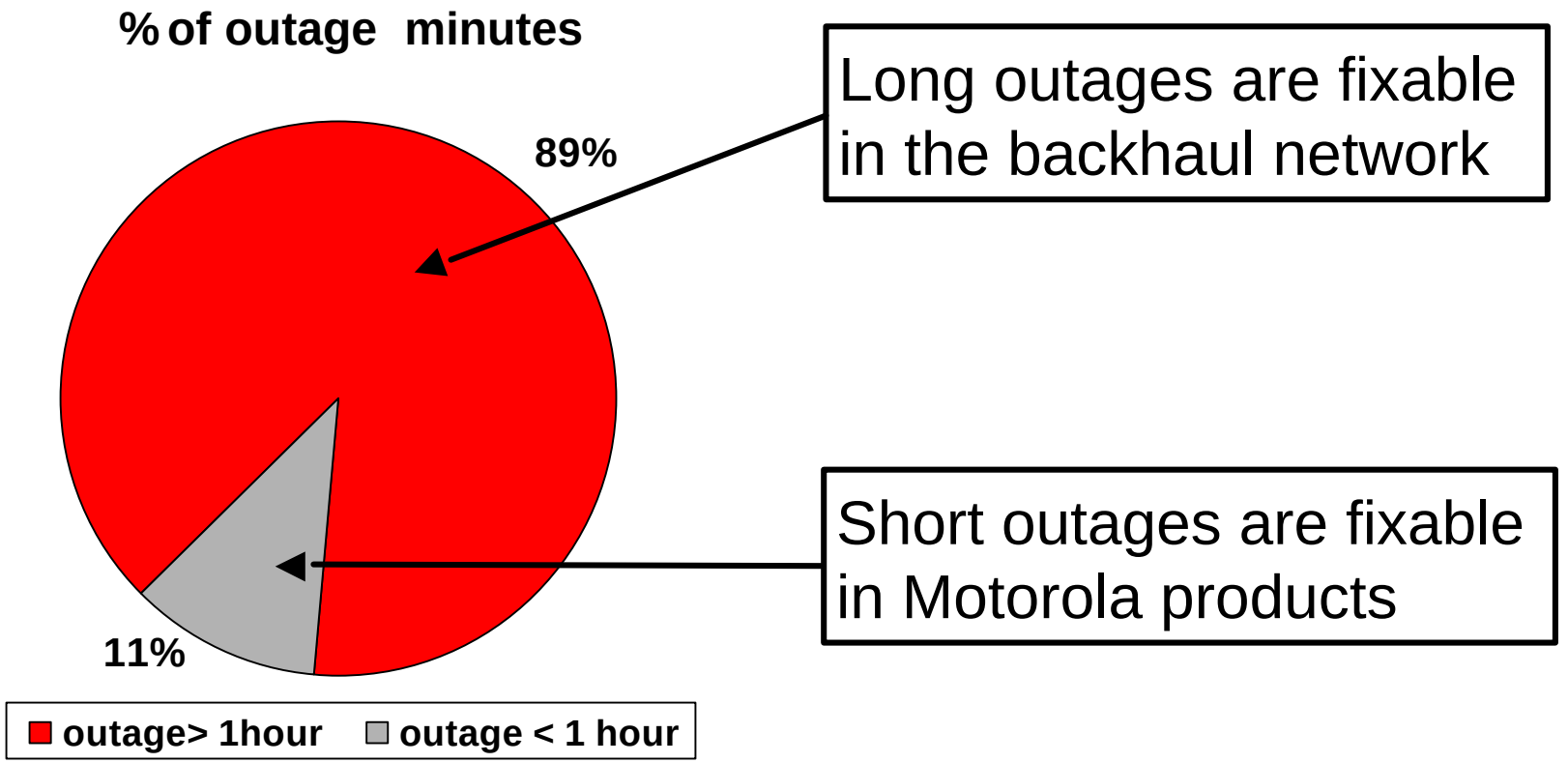
intelligence everywhere™

Breakdown of outages > 1 hour (all in the backhaul network)



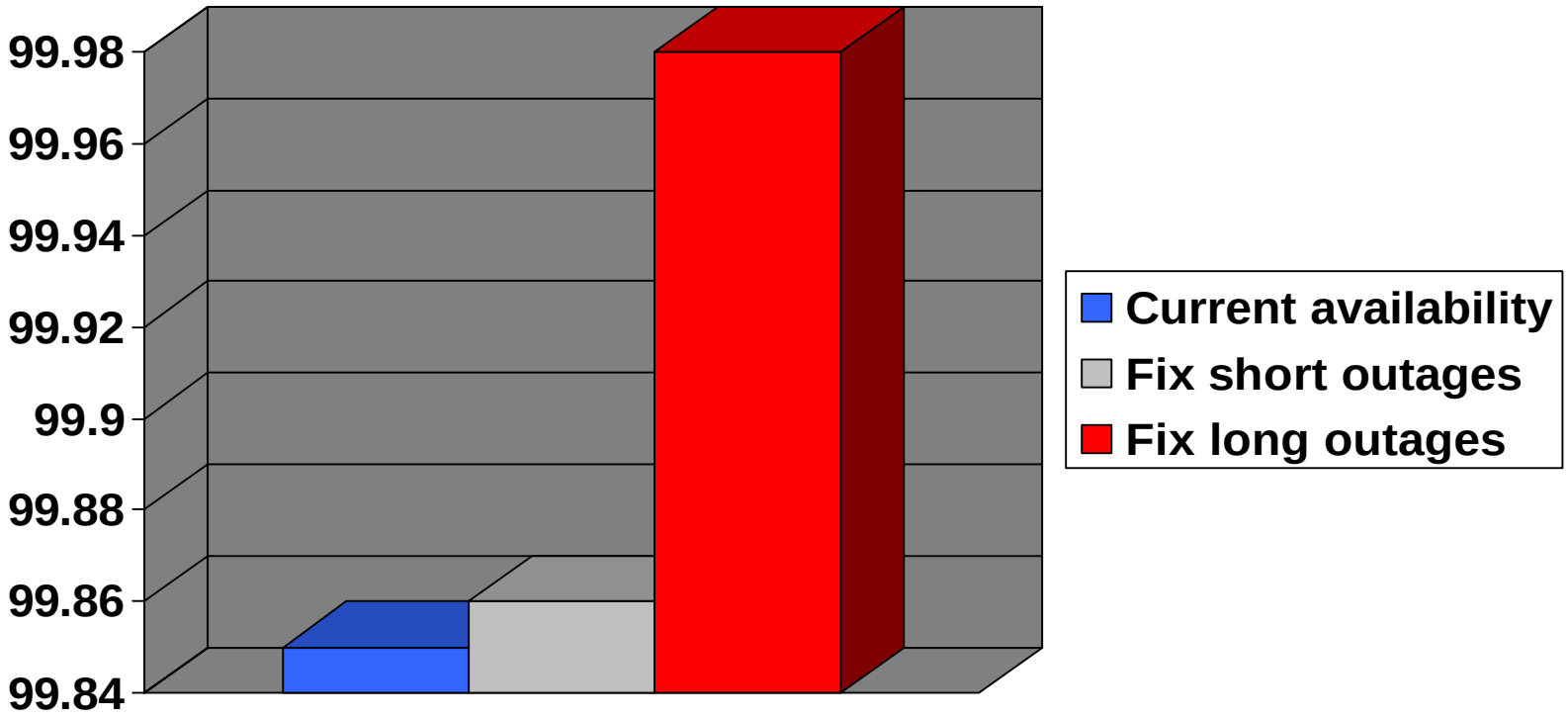
intelligence everywhere™

Where problems can be fixed



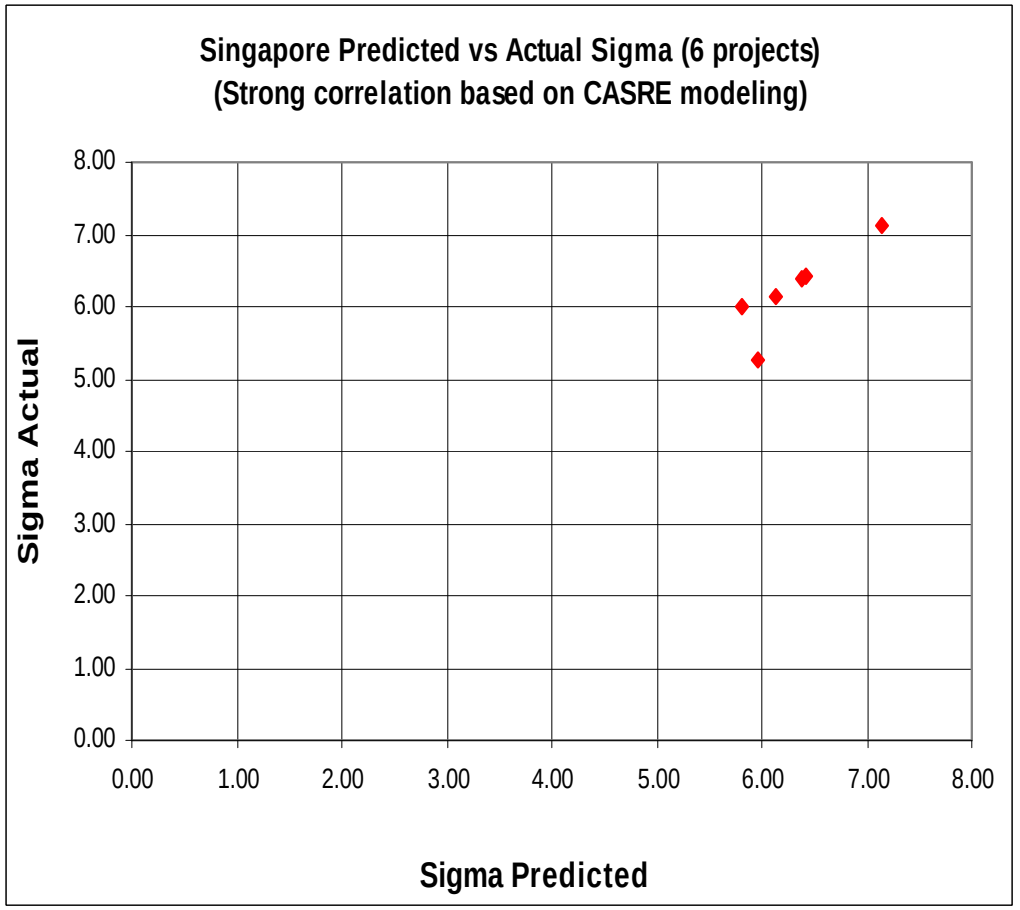
intelligence everywhere™

Impact of fixing outage classes on availability



intelligence everywhere™

Example of SRE Use in GSG Centers



Key Academia Challenges to Improve Software Reliability

- **Apply existing statistical models to real-world software environments**
- **Enhance SRE models and approaches to cover systems and products**
- **Develop more robust SRE models and techniques**
- **Expand SRE curricula to match industry needs**

Key Academia Challenges to Improve Software Reliability

- **Apply existing statistical models to real-world software environments**
  - Form collaborations with industry to analyze project data
  - Have students and professors intern with software industry
  - Develop systematic approaches for defining operational profiles
- **Enhance SRE models and approaches to cover systems and products**
  - Create mixed or hybrid models including software and hardware
  - Develop SRE models incorporating COTS and 3rd party software
  - Create composable component-based software reliability models
- **Develop more robust SRE models and techniques**
  - Allow for limited or incrementally improving customer information
  - Account for operational profiles that change with use over time
- **Expand SRE curricula to match industry needs**
  - Include interpersonal & change agent skills plus business knowledge
  - Include system engineering along with software and statistics

Critical Need for Software / System Reliability Engineering Today:

Integrated and interactive systems that help industry design, simulate, and validate technology- and product-independent customer-oriented high-availability system solutions based on hardware, software, and user interactions – to achieve 5 NINES in use (99.999% available).

- **We need to identify acceptable customer tolerances / tradeoffs for …**
  - **Higher quality**
  - **On-demand time to market**
  - **Low cost maintenance**
- **We need to focus on the following key items:**
  - **Decomposing requirements & simulating critical system elements based on systematically developed operational profiles**
  - **Aggregating and designing for system reliability based on component reliability data, incorporating hardware, software, and end users**
  - **Analyzing system behavior by operating on simulated and measured data**
  - **Leveraging advanced testing & analysis techniques to maximize availability**

Software and System Reliability – the Next 10 Years

Weird animals surround me in my home, at work, everywhere I go. Every day I must spend hours feeding them, healing them, waiting for them. And the fighting! They hold each other hostage in asphyxiating headlocks. I scream at them, but they just grunt or stare back stupidly. When we do get along, and I'm feeling affection for them, they suddenly turn around and bite a chunk off my hide.

You are surrounded by these creatures, too -- the personal computers, laptops, handheld assistants, printers, Internet-savvy phones, music storage drives, and other digital wonders. They are everywhere and multiplying fast. **Yet instead of serving us, we are serving them. We wait endlessly for our computers to boot up, and for bulky Web pages to paint themselves on our screens. We stand perplexed in front of incomprehensible system messages, and wait in frustration on the phone for computerized assistance. We constantly add software upgrades, enter odd instructions, fix glitches, only to sit in maddening silence when our machines crash, forcing us to start all over again, hoping against hope that they didn't take a piece of our intellectual hide with them.** We'd never live in a house, work in an office, or ride in a car where we had to put up with a menagerie of such beasts. Yet we do it every day with our computer menagerie.

We shouldn't have to.

Excerpt from The Unfinished Revolution by Michael L. Dertouzos

Software and System Reliability – the Next 10 Years:
Platform-Independent, Interoperable, Context-Aware Systems

- **From millions of connected systems to billions and beyond**
  - Wireless and wired intelligent appliances – cell phones, PDAs, …
  - Intelligent infrastructure – rooms, equipment, clothing, …
- **From machine-centered to human-centered software and systems**
  - Personalized, always-accessible information and experience
  - Portable software that negotiates with its environment
  - Human-controlled interactions and information security
- **The future Reliability/Availability Grand Challenge**
  - Constantly available software and information
  - Transparently-interoperable processing capability
  - Adaptable systems that accept new intelligent environments
  - The ability to run perfectly anywhere at any time on any platform