

Methods, Processes & Tools for Global Software Development



Dan Paulish
daniel.paulish@siemens.com



Agenda

Motivation for Global Software Development
(GSD)

Challenges

An Approach

Lessons Learned

Open Issues and Research Questions

Global Software Development (GSD): Motivation

Lower development costs

- *Low-wage countries*

Capitalizing on a pool of trained workforce

- *Quest for talent*

Increased output, reduced time

- *Improve time-to-market*
- *Round-the-clock development*

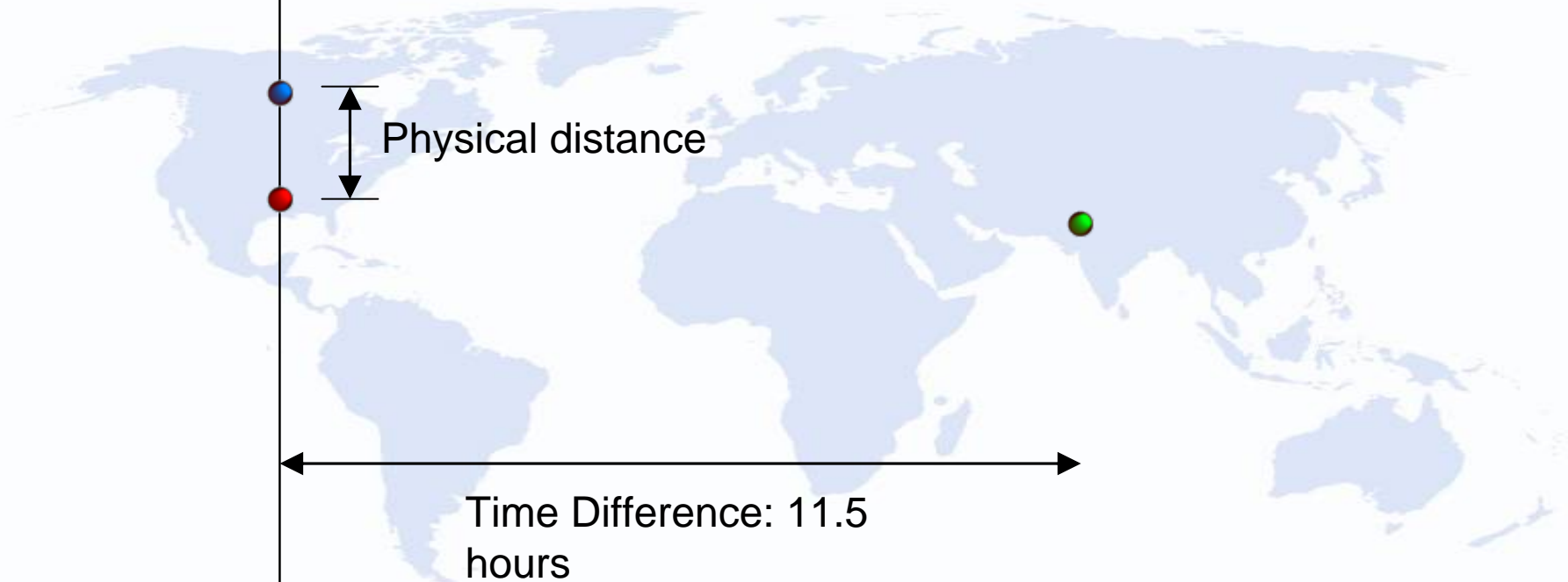
Market proximity

- *Specific local expertise*
- *Market acquisition effort*

Governmental policies and incentives

Original motivation
was reduced
development cost,
but there are other
reasons for GSD.

But, Global Software Development is Difficult.

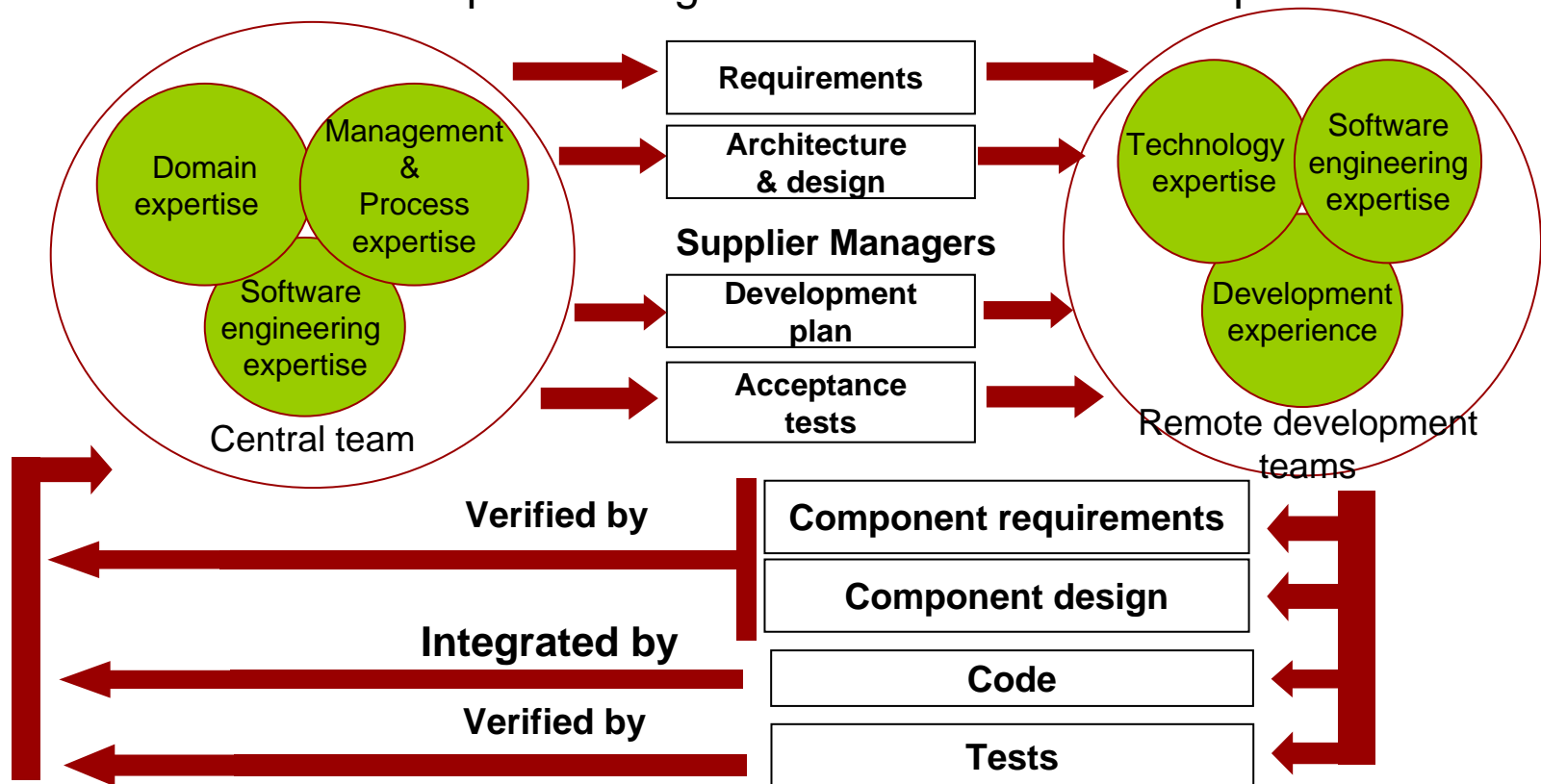


Global Software Development

Siemens has more than 15 years experience developing software products globally.

Today, most Siemens business groups develop software in low-cost countries.

Organization and process model for global development divides responsibility between a central headquarters organization & remote development teams.

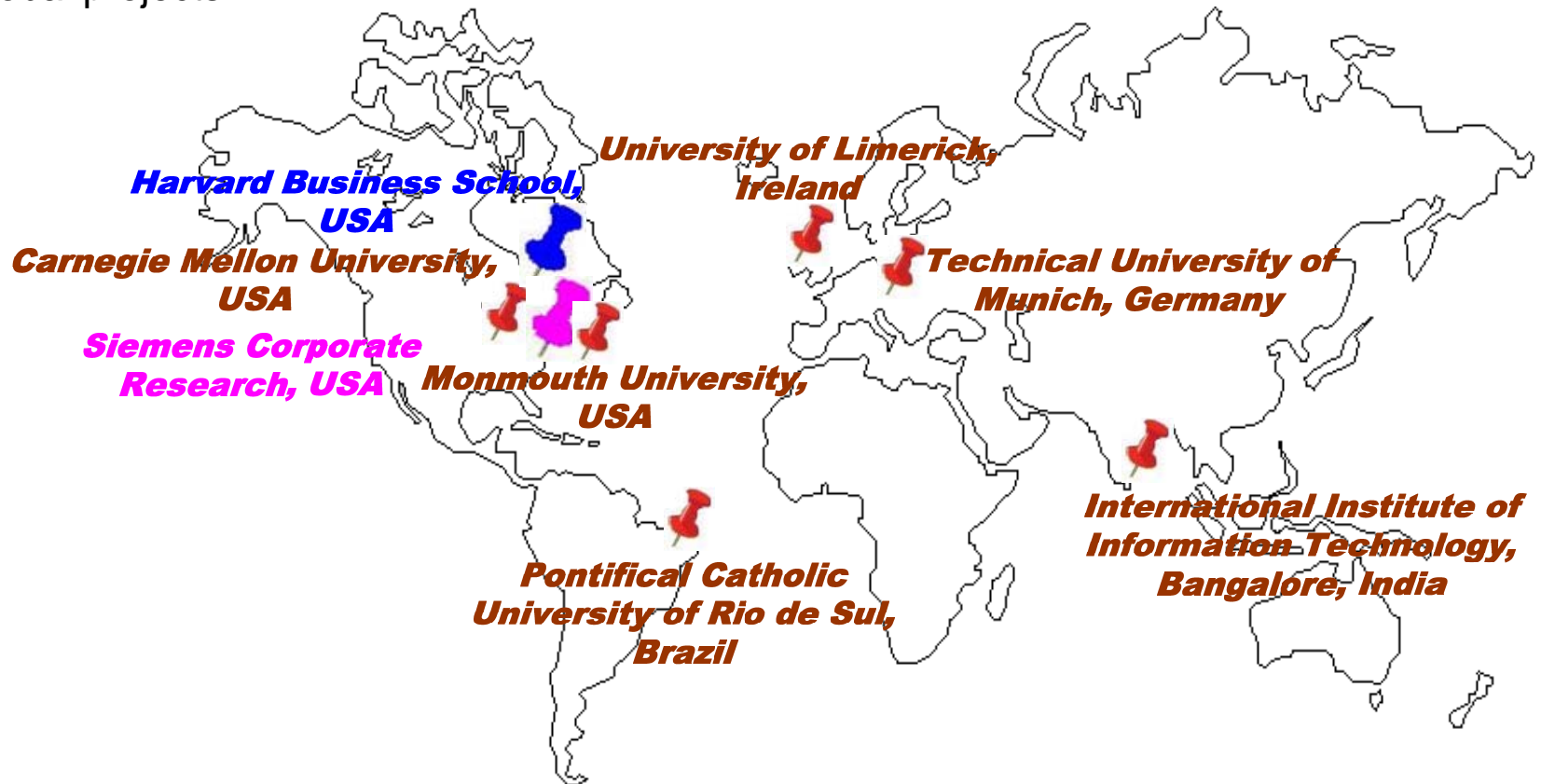


Global Studio Project (GSP)

Experimental global software development project using university student teams and researchers.

Shadows real Siemens global development project, process, & organization.

Document processes, best practices, and understanding of how to successfully execute global projects.



GSP Research Goals

Identify and document **best (and worst) practices** for global software development.

Identify the **prerequisites** for **successful global software development**.

Test a global software development **reference process**.

Determine **artifacts** for **commissioning a remote development team**.

Identify **communications necessary** for effective global development.

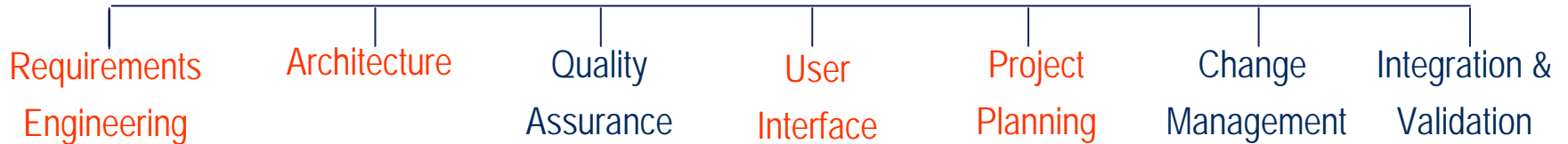
Example Organization – Global Development Project

Product Line

Centralized Roles:

Manager

Chief Architect



Application Development:
Distributed Synchronized Teams

R&D

Resource Manager

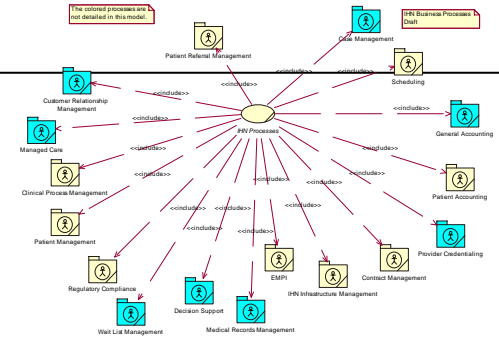


QA, architect, and subject matter experts are embedded within the small component teams.
Component teams use agile processes and are synchronized by the centralized team.

Centralized Functions Product Line Management

Component Development

Requirements Engineering

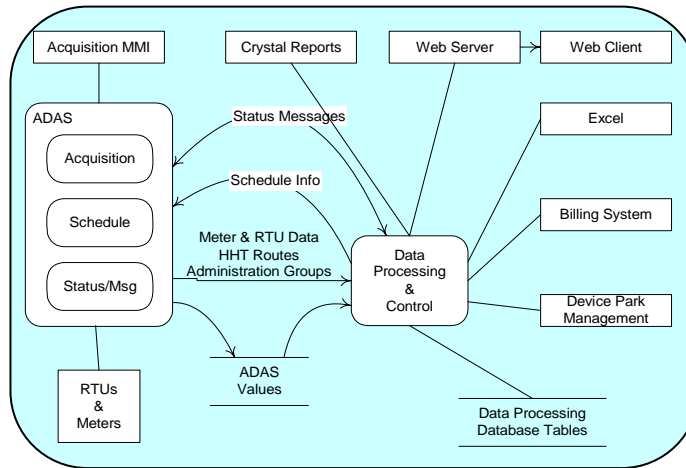


User Interface Design

Architecture Design

System Integration & Validation

Quality Assurance



Project Planning

	ER1	ER2	ER3	R1	R2+
Schedule Maker					
Search Consumer Tree for Scheduled Events	√			√	
Create a Schedule	√			√	
Handle Report Events	√			√	
Handle Acquisition Events		√		√	
Optimize Acquisitions				√	
Handle set Parameter Scheduled Events		√		√	
Display and manual Update of Schedules			√	√	
					√

Remote Development Functions

Application Component Development teams are given:

- Part of business model to implement
- Acceptance tests
- Incremental development plan and integration dates
- Component interface specifications
- Vertical slice implementation
- UI Style Guide

Development Team Roles:

- Project Leader
- Subject Matter Expert
- Architect
- Developer
- QA/Component Testing



Experts at each development site

Global Studio Project Development

MSLite Project:

- MSLite: Management Station Lite
Building Automation domain
- Shadows a real Siemens distributed development project

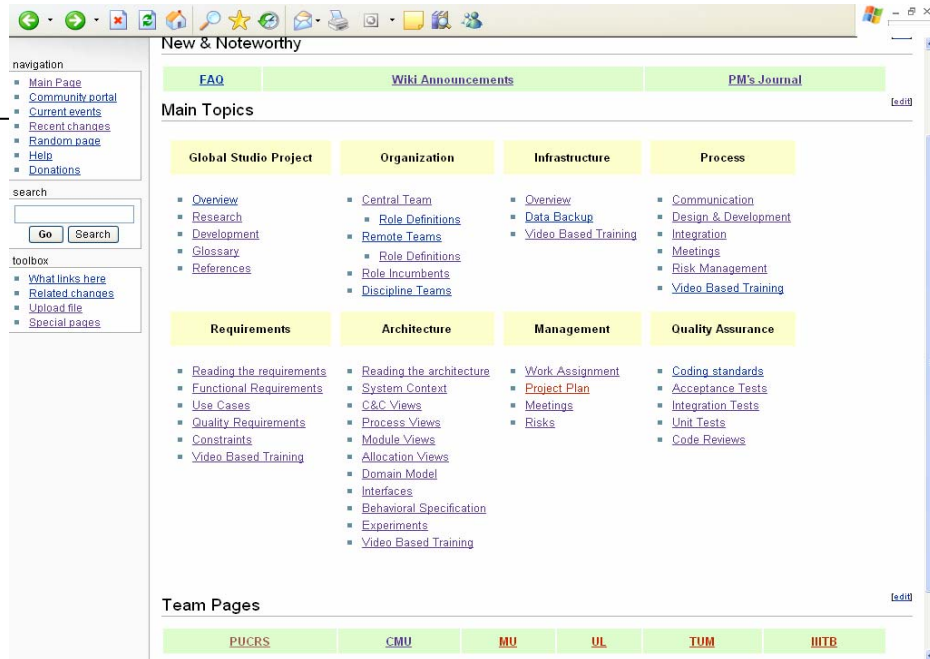
Global Studio Project Size

7	Teams
30	Developers
5	Countries
4	Continents
11	Time Zones
67.5	Delivered KLOCs

Key Practices

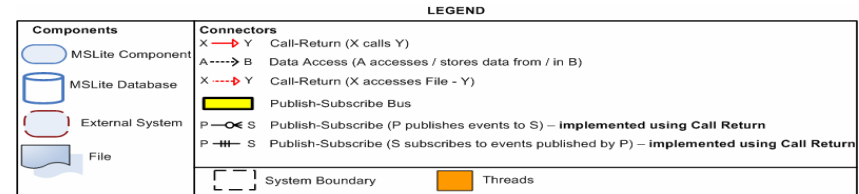
- Architecture-Centric Approach
- Wiki-based Collaboration/Content Management System
- Common Development Environment & Tools/Continuous Integration
- Social Network Analysis (SNA)

Practical Approach



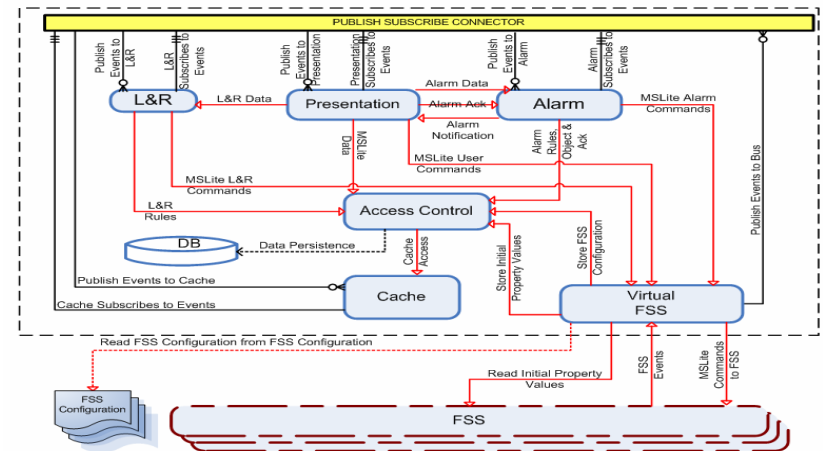
Wiki-based collaboration

- Central artifact repository
- Central team & remote team pages
- Templates and organization controlled by central team
- Remote teams complete documentation tasks on the wiki



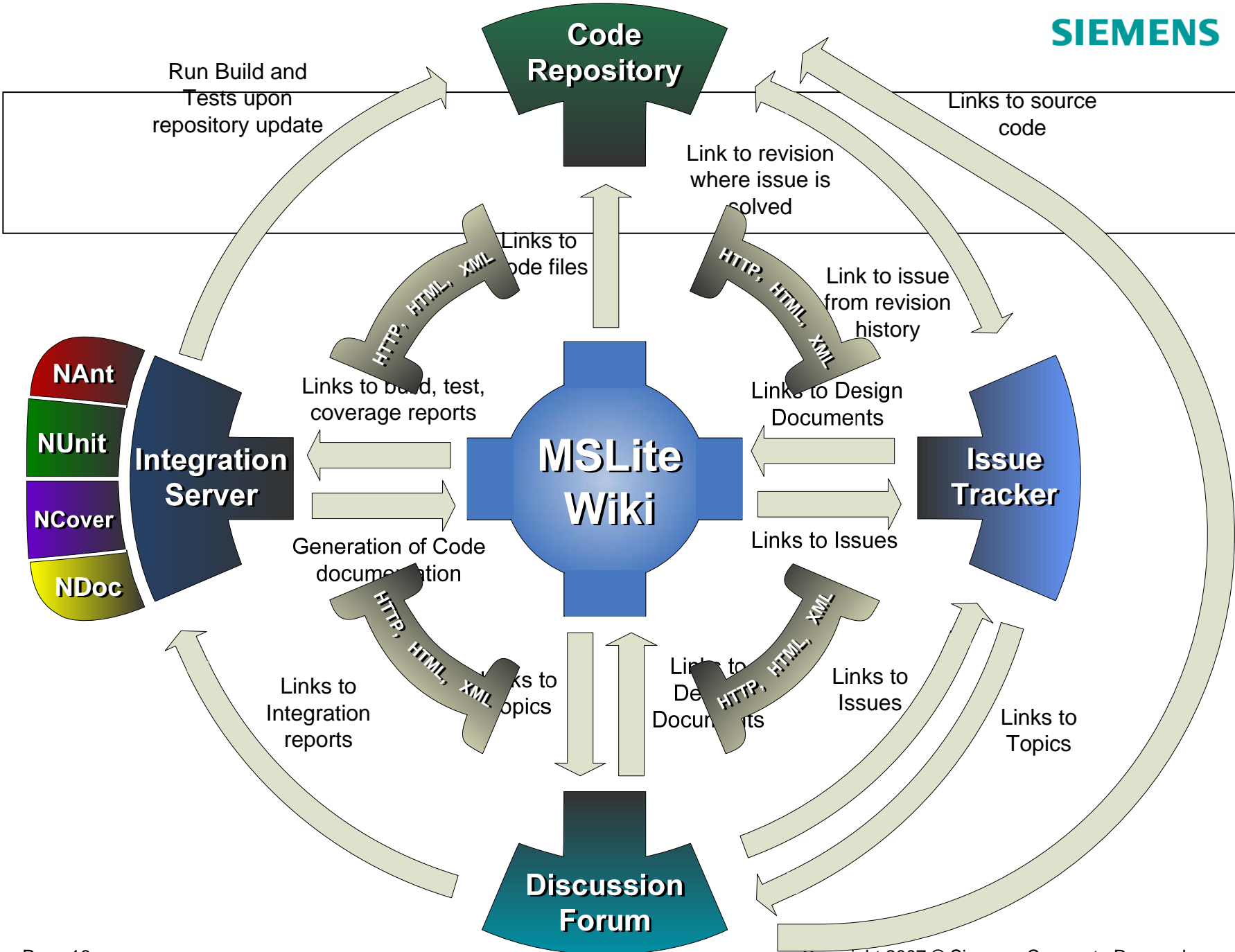
Architecture-centric approach

- Multiple architectural views
- Clear division between central and remote team responsibilities
- Clear (and navigable) traceability between artifacts (e.g., requirements-design-test)
- Project plan based on architecture



Development Environment

- Teams involved in requirements, design & development workflows during Elaboration, Construction, Transition phases
- Component-based decomposition and distribution of development effort
- Continuous integration tools
 - Automated builds
 - Automated tests, metrics, API documentation
- No uncoordinated inter-team communication, everything is channeled through the central team
- Tools for direct/indirect communication (Wiki, Issue Tracker, Mailing Lists)



Communication Infrastructure -1

- Weekly teleconferences
 - Quick problem resolution
- Videoconferences and Face-to-Face meetings
 - Humanize perception of remote parties
 - Establish trust relationship
- Team Mailing Lists
 - Active Notification

Communication Infrastructure - 2

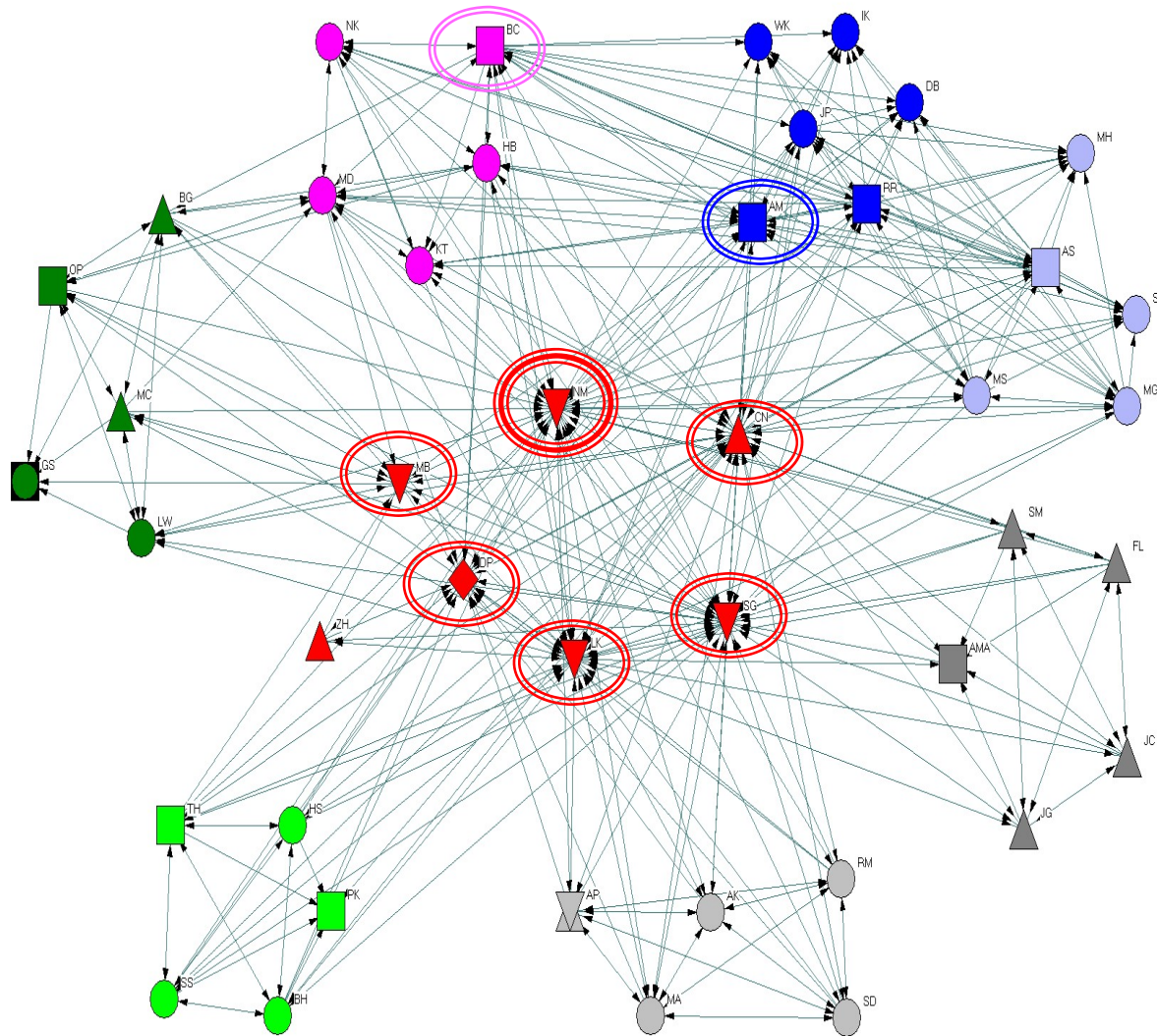
Collaborative Content Management System

- Organizational Information
- Requirements: Functional, Use Cases, UI Mockups
- Architectural Views, Interface Specifications
- Project Planning and Tracking
- Process Reference

Dedicated Tools

- Defect Management
- Change Management
- Code Versioning
- Content Discussion

Social Network Analysis (SNA)



Affiliations (Sites)

- SCR Central Team
- Tech. Univ. Munich
- CMU Sapphire
- IIITB
- Monmouth #5
- Monmouth Codicons
- Monmouth TCT
- Limerick ArdnaCroise

Roles

- Developer
- Team Leader
- Function Manager
- Remote Supp. Manager
- Central Supp. Manager
- Executive

Some Lessons Learned

- Continuous Integration
- Amount of Required Communications
- Cultural Differences among Teams
- Start-up Time for Remote Teams
- Supplier Managers as the interface to Remote Teams
- Team Size
- Agility

Open Issues and Research Questions

1. Given that the technical artifacts that are delivered to the remote component development teams are not adequate in specifying the precise work to be done, in what ways are they deficient?
2. What strategies do the remote teams employ to compensate for the deficiencies found in the received technical artifacts?
3. What are the early warning signs that an issue is imminent? Can communication patterns, for example, between the central and remote teams be used to predict future component integration problems?

Thank you!



Contact:

Dan Paulish

Distinguished Member of Technical Staff

daniel.paulish@siemens.com

Phone +1 (609) 734-6579