Service-Oriented Architecture for Command and Control Systems with Dynamic Reconfiguration

Raymond A. Paul

Department of Defense
Washington, DC
raymond.paul@osd.mil

Outlines

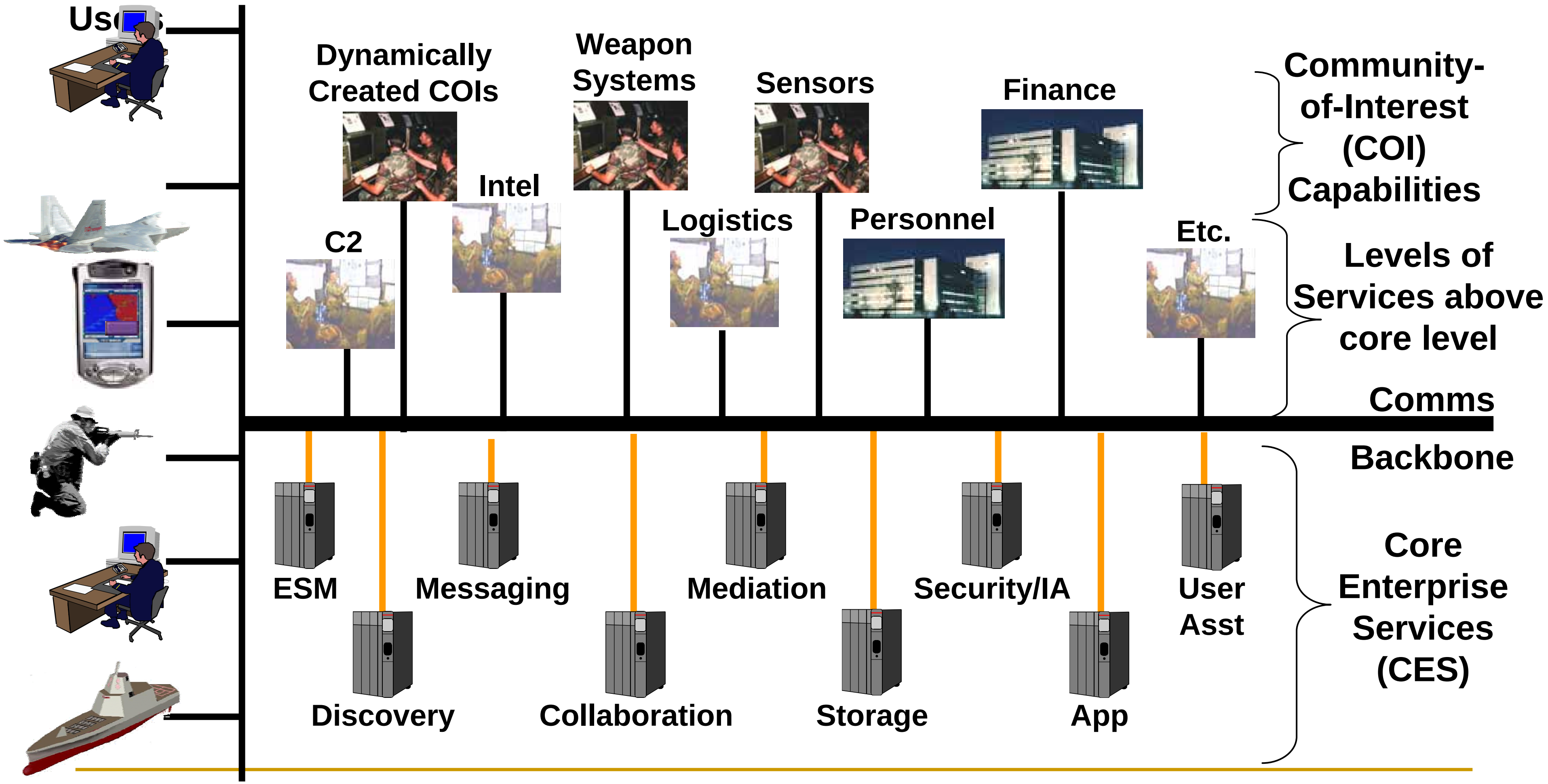
- Motivation
- Dynamic Reconfigurable C2 System Model & Requirements
- Features of C2 Dynamic Reconfiguration System
- Architecture of C2 Dynamic Reconfiguration
- C2 Policy Specification and Execution Language
- Conclusion and Future Work
- Appendix: Prototype Illustration

Motivation

- Command and Control (C2) systems are moving into SOA as evidenced by the recent development of NCES and GES (GIG Enterprise Services).
- C2 system may need to be compatible with NCES and GES.
- Modern warfighting needs a dynamic, adaptable and agile force supported by rapidly changing technology.

NCES Vision

Support real-time & near-real-time warrior needs and business users



Users

Dynamically Created COIs

Weapon Systems

Sensors

Finance

Intel

C2

Logistics

Personnel

Etc.

Community-of-Interest (COI) Capabilities

Levels of Services above core level

Comms

Backbone

ESM

Messaging

Mediation

Security/IA

User Asst

Discovery

Collaboration

Storage

App

Core Enterprise Services (CES)

2004-5-22

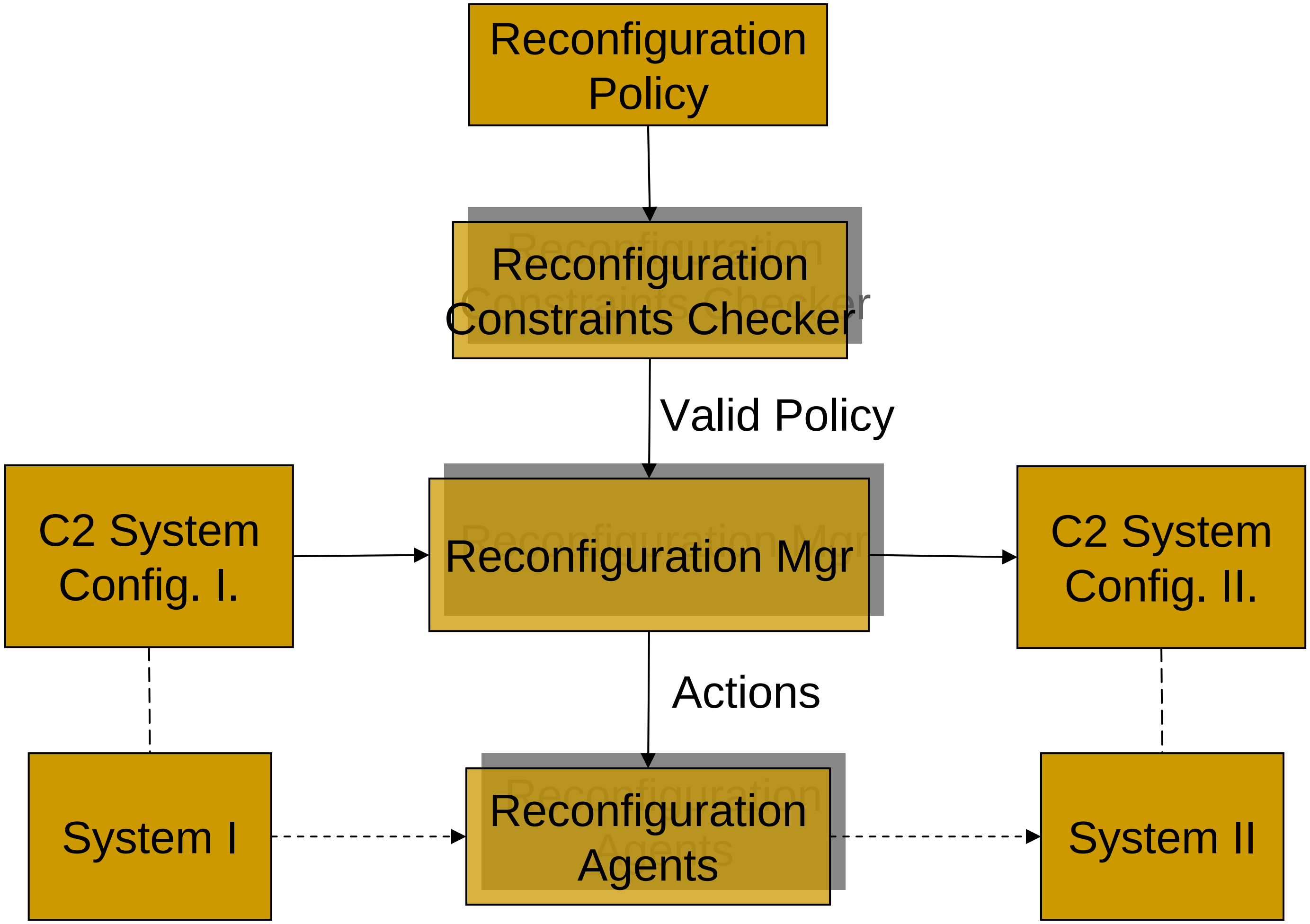
Service-Oriented Architecture (SOA)

- System interacts with each other with standard protocols such as SOAP, UDDI and WSDL;
- Each computation unit is a service and its specification is published in a WSDL file;
- Service lookup, searching, binding are done at runtime by the UDDI server;
- New services can be added into the system without changing the overall system architecture
- The enterprise C2 system can easily interoperate with numerous existing systems including weapon systems, communications, sensor systems, and other C2 systems for commanders.

Limitation of Current SOA for C2

- However, current SOA cannot satisfy the survivability requirements for mission-critical C2 systems. It fails to discover and rebind to available services automatically upon system failures or overload.
- C2 systems will be subjected to various attacks including physical attacks as well as electronic attacks.
- One key feature of survivability is that the C2 system must be able to dynamically reconfigure its participating subsystems and continue its intended operations in case of failures or overload.

Dynamic Reconfiguration Model for SOA



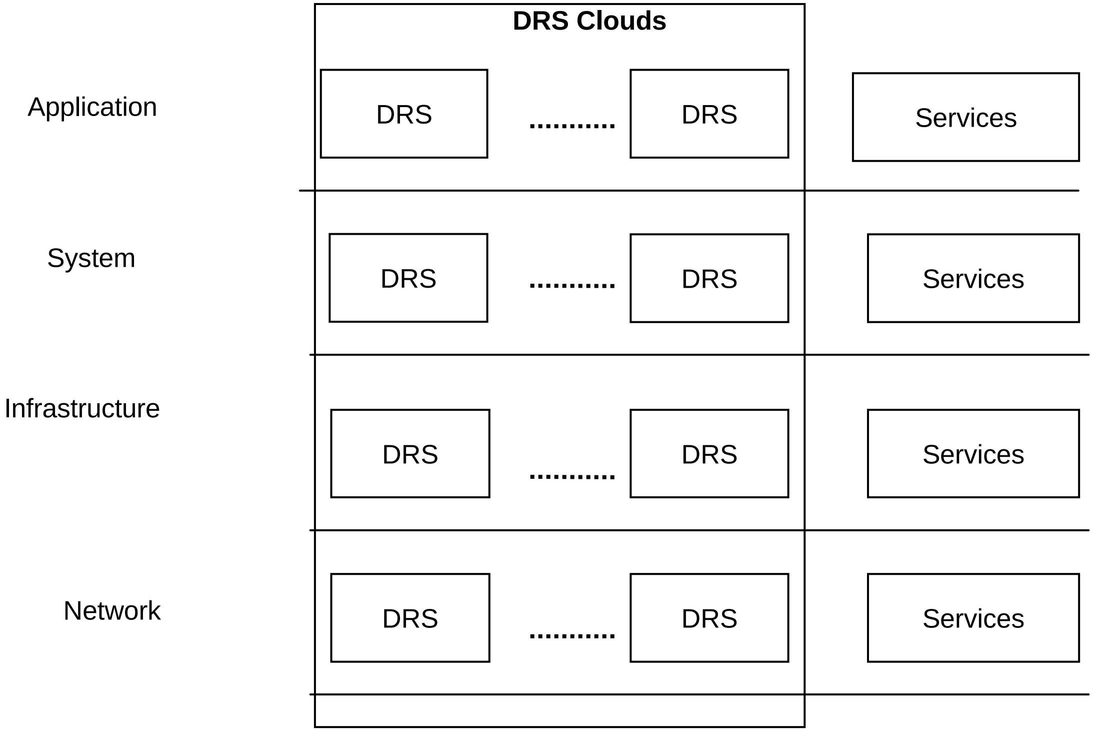
Requirements for the distributed dynamic reconfiguration.

- Efficiency: the reconfiguration algorithm should introduce minimum disruption to the system
- Consistency: the reconfiguration action should maintain the consistency of related components after reconfiguration.
- Correctness: Reconfiguration constraints must be satisfied at all the times.
- Real time: Reconfiguration must be carried out in real time at runtime for mission-critical C2 system.
  - Real time means within a time limit
  - Runtime means that the reconfiguration must be carried out while the system is still in operation.

Requirements for the distributed dynamic reconfiguration (cont')

- Policy driven: C2 system is governed by C2 policy, possibly distributed policies.
- COI based: C2 is monitored and controlled by the participating COIs.
- Distributed and parallel execution.
  - Reconfiguration actions are executed by the distributed agents and the reconfiguration actions are done in parallel among these agents.
- Service-oriented
  - The dynamic reconfiguration server or agents are themselves services in the SOA, so that they can be reconfigured too in case of their own failures.
- Dynamic Reconfiguration with hierarchical and layered C2 system
  - This promotes survivability

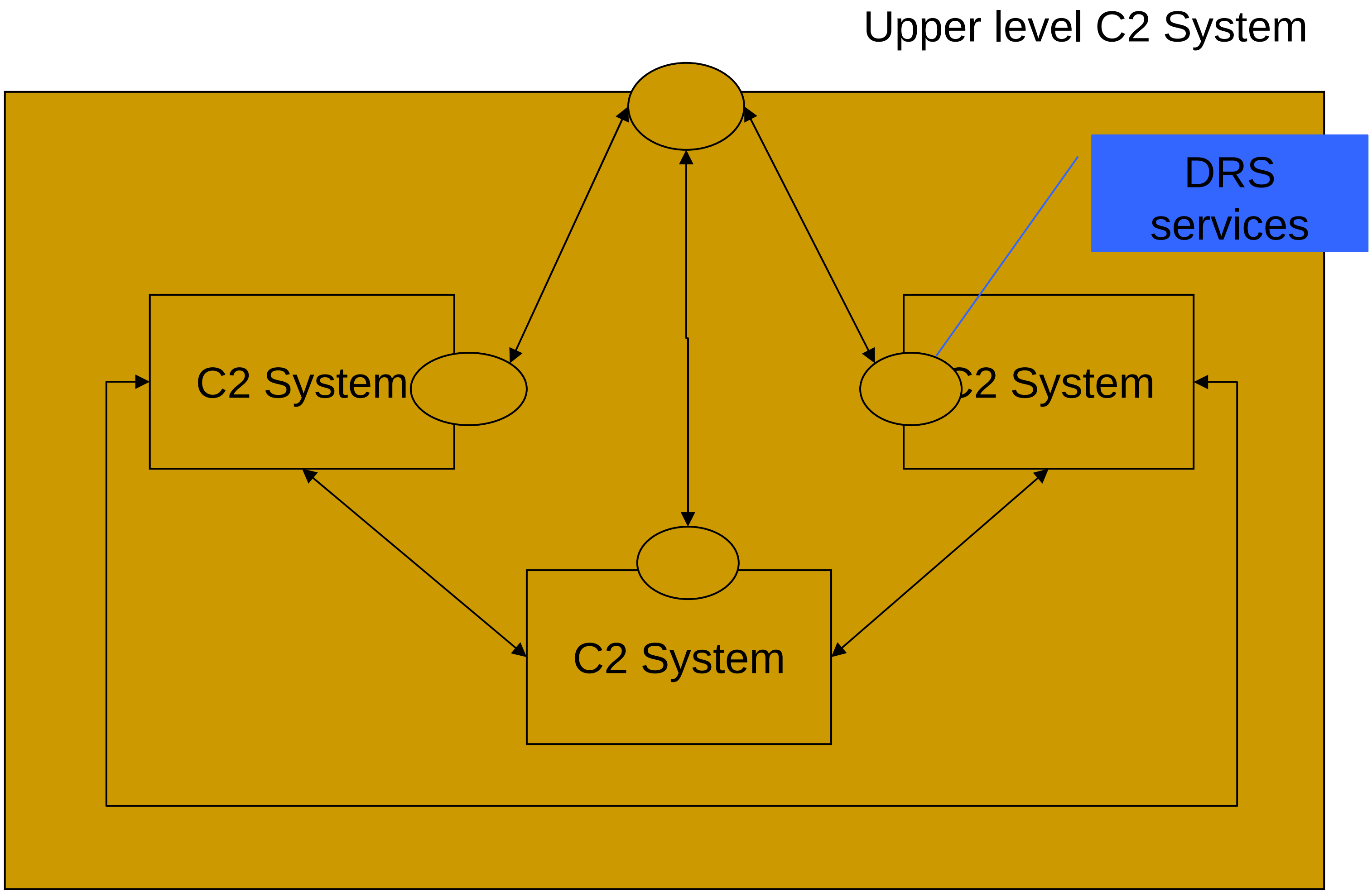
Layered Dynamic Reconfiguration Architecture



Layered Dynamic Reconfiguration Architecture (cont')

- Dynamic Reconfigurability is embedded in each layer of the entire system, and at each level multiple DRS synchronized with each other to avoid any single point of failure.

Hierarchic C2 Dynamic Reconfiguration



Hierarchic C2 Dynamic Reconfiguration

- SOADR (SOA with Dynamic Reconfiguration) allows the distributed system to be constructed in a hierarchical manner.
- A composite system can be constructed from primitive systems with dynamic reconfigurability and these in turns can be constructed into more complex composite system with dynamic reconfigurability.

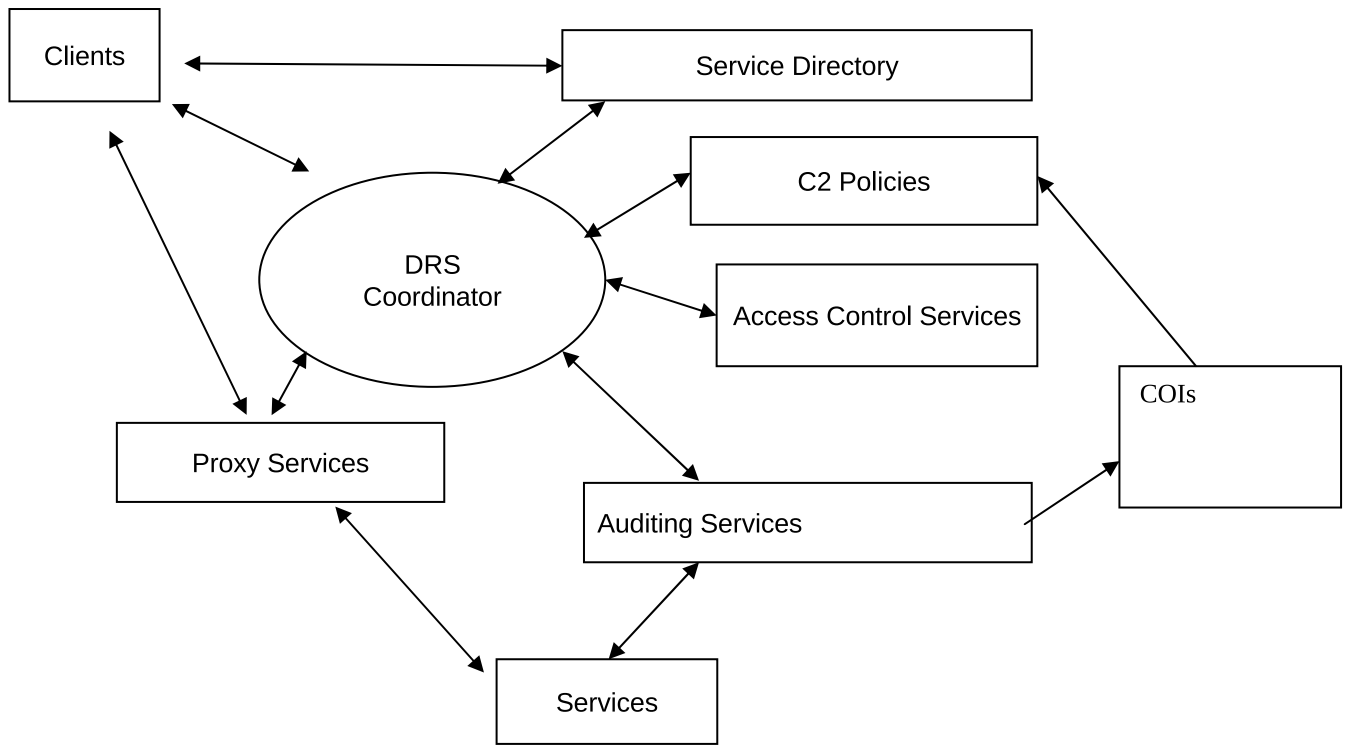
SOA with Dynamic Reconfiguration

- Every participating C2 unit, including DRS, is a service and each service is treated the same. As a service, the DRS provides the following functions:
  - Dynamic service lookup, service publication, service binding, and service profiling,
  - Registration and de-registration,
  - Runtime services verification including constraint verification such as security verification, interoperability checking, and performance monitoring.

SOA with Dynamic Reconfiguration (cont')

- The Dynamic Reconfiguration Service (DRS) is implemented as a critical service with redundancy and the reconfiguration strategies and algorithms can be changed at runtime to fit the warfighting needs.
- With this kind of mechanism, the behavior of the SOA can be changed in real time at runtime, and even the behavior of DRS can be changed too.

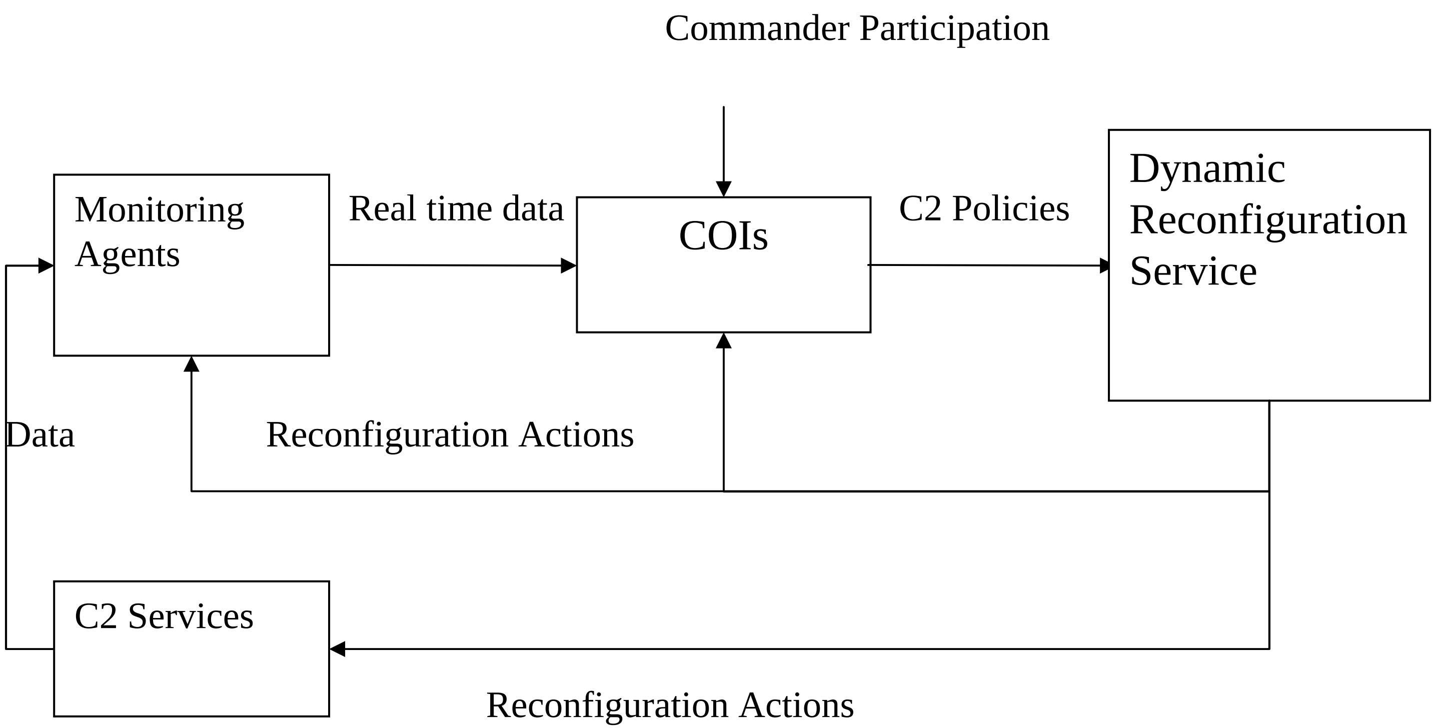
Architecture of DRS and its Distributed Agents



Architecture of DRS (cont') Major Agents

- Service Directory (SD): This stores and organizes services in a hierarchical tree with internal tree node representing a group of related services.
- Proxy Agents: An Proxy Agent (PA) is responsible for interoperability and integration between DRS, services and their clients. In addition, it also enforces security accessing control.
- Auditing Agents: An Audit Agent (AA) monitors and checks the performance and user concerned properties of the participating services at runtime and updates their profiles.

Cyclic Flow of Dynamic Reconfiguration Process

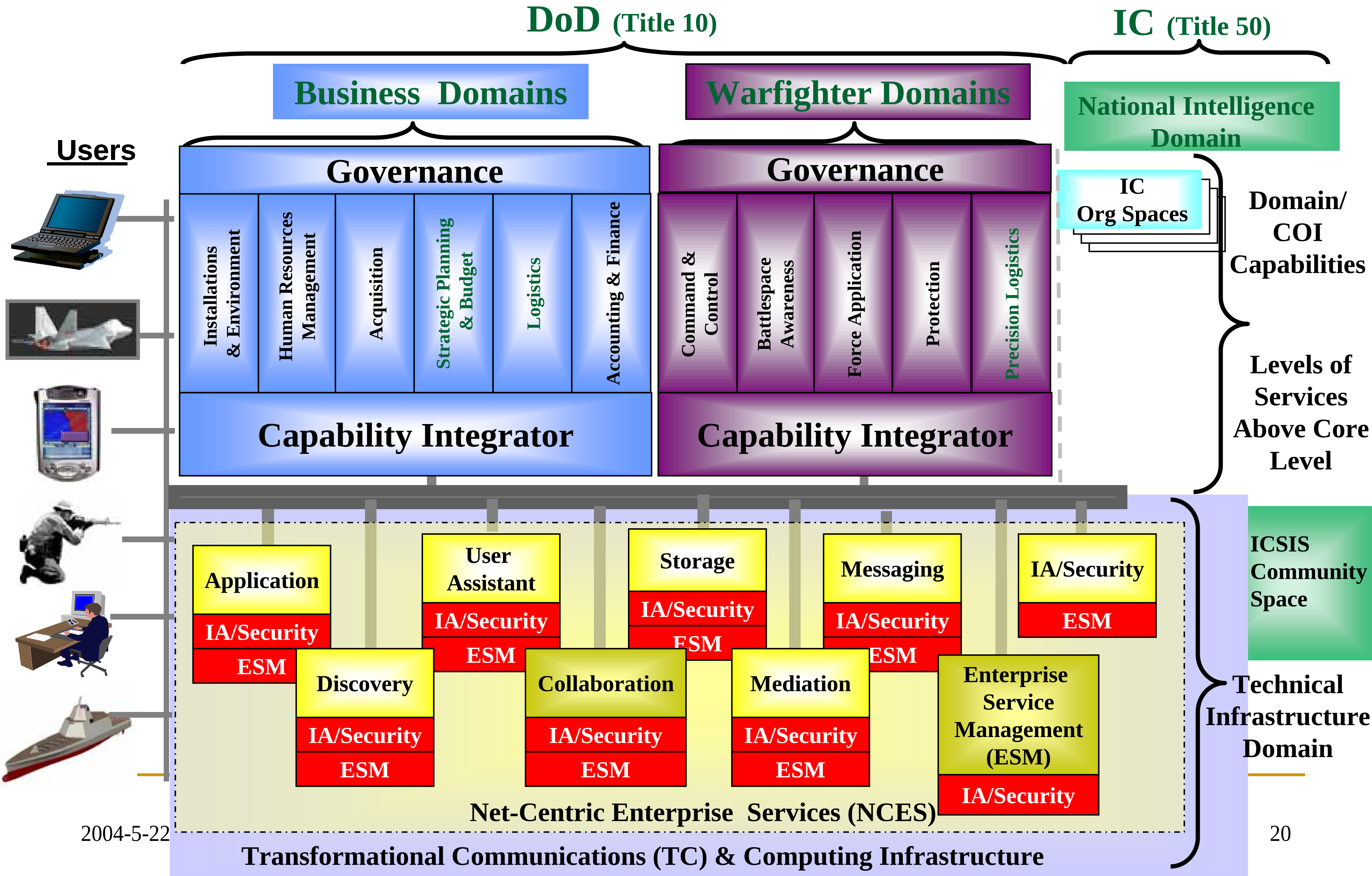


Dynamic Policy Adaptation and Validation

- Dynamic Policy Adaptation
  - Through the cyclic flow of situation aware of environment data collect and monitoring, reconfiguration policy formation, policy validation and reconfiguration execution.
  - Commander in the loop
- Policy Validation
  - Reconfiguration Constraints are represented by Meta C2 Policy and it will validate the reconfiguration policy in real time at runtime

GIG Enterprise Services

SOADR Supports real-time & near-real-time warrior needs



DoD (Title 10)

IC (Title 50)

Business Domains

Warfighter Domains

National Intelligence Domain

Users

Governance

Installations & Environment

Human Resource Management

Acquisition

Strategic Planning & Budget

Logistics

Accounting & Finance

Capability Integrator

Governance

Command & Control

Battlespace Awareness

Force Application

Protection

Precision Logistics

Capability Integrator

IC Org Spaces

Domain/ COI Capabilities

Levels of Services Above Core Level

Application

IA/Security

ESM

User Assistant

IA/Security

ESM

Storage

IA/Security

ESM

Messaging

IA/Security

ESM

IA/Security

ESM

Discovery

IA/Security

ESM

Collaboration

IA/Security

ESM

Mediation

IA/Security

ESM

Enterprise Service Management (ESM)

IA/Security

ICSIS Community Space

Technical Infrastructure Domain

Net-Centric Enterprise Services (NCES)

Transformational Communications (TC) & Computing Infrastructure

Cyclic Flow of Dynamic Reconfiguration Process (cont')

- The dynamic reconfiguration mechanism is controlled by a set of C2 policies, and these policies specify appropriate actions to take under various situations.
- These C2 policies can be pre-specified before system operations, but also they can be updated during runtime in real-time by the COI (community of interest) within the C2 system.
  - For example, both NCES and GIG have real-time COIs that can be used to determine and construct C2 policies based on data collected from situation-aware monitoring agents or services.
- The distributed situation-aware COI monitoring agent detects the changes in the operation environment, it informs the concerned COIs and then these COIs may update their C2 polices to adjust to the changing environment.

Cyclic Flow of Dynamic Reconfiguration Process (cont')

- Once the C2 policies are changed, the dynamic reconfiguration is changed as it is ruled by the C2 policies, even though the overall reconfiguration mechanism remains the same.
- In this way, commanders and decision makers make high-level decisions, based on the latest situation assessment, and let the SOA to actually implementation the transition plan.
- Multiple monitoring agents, COIs, and DRS can participate in this process, and this process is done in a distributed but collaborative manner.

Policy Specification & Execution Language (PSEL)

PSEL is designed to

- Support policy-driven computing in GIG as it has Policy-Based Networking and Common Open Policy Service.
- Specify service constraints and dynamic reconfiguration polices and can be easily integrated with C2 services and environments.
- Be well formed thus facilitate policy specification, enforcement and revision.

Policy Enforcement by Trusted Agents

- Once the dynamic reconfiguration C2 Policy are defined, they can be used to verify and audit various system properties at runtime and governs the reconfiguration process.
- It is enforced by distributed trust agents.

Sample C2 Reconfiguration Policy with COI

Sample

- **Policy**: In case of the JFC system fails, move the position location task to AWY system. If AWY system is overloaded, then move the position location task to the least busy system runtime detected.
- **Specification**:

require “position location task” **to** Move **to** AWY **when** (JFC.Status = “fail” || AWY.status = “normal”)

require “postion location task” **to** Move **to** System decided by COI *at runtime* **when** (JFC.Status = “fail” || AWY.status = “busy”)

Conclusion

- This paper proposed a SOA with dynamic reconfiguration for the DoD C2 system based on an extension to the existing WS architecture.
- In this framework, every process is treated as a service, either an atomic service or a composite service.
- The extensions include service specification, policy specification and execution, runtime monitoring and verification, and dynamic reconfiguration. A prototype tool has been implemented to illustrate these concepts.

Conclusion

- The proposed SOADR addresses survivability of C2 systems as each process or services running in the framework is continuously monitored by at least one service, and any failure or overload for the particular service will trigger dynamic reconfiguration in real time at runtime under the control of C2 policies without human intervention.
- The proposed SOADR also addresses rapid development and evaluation. As any new software can be developed as a service, and once it is placed in the SOADR, it can be picked up by other applications running on top of SOADR.

Future Work

- Model-driven dynamic reconfiguration
  - Embedded C2 system, NECS and GES with Dynamic Reconfigurability
    - Once the NECS or GES system is specified, the deployed system automatically has the capability of dynamic reconfiguration.
- Policy-driven environment detection and monitoring framework
- Reconfiguration execution framework

Illustration

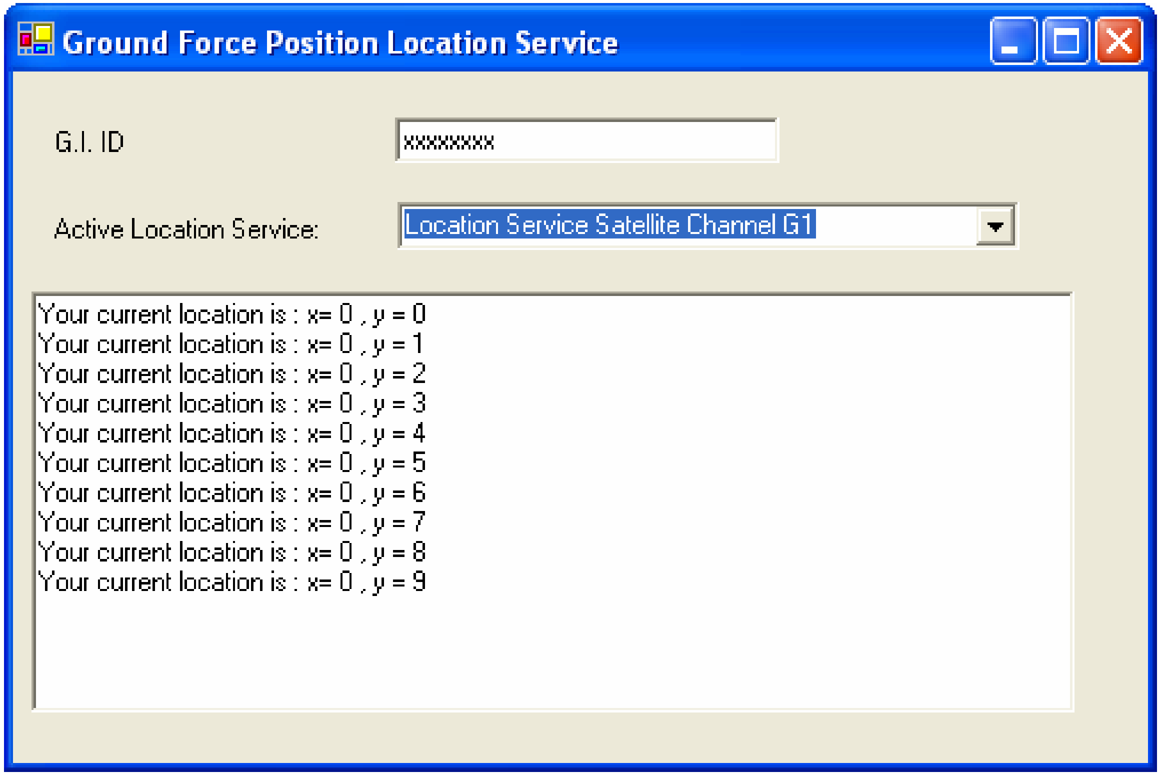
In the prototype, we illustrate how the system reacts based on the reconfiguration policy. The following slides shows, when the monitor agent detects the failure of service to channel G1, The SOADR switches to the location service to Channel G2. The current SOADR is implemented on the .NET platform.

Location Service Reconfiguration Policy

- **Policy**: In case of the Location service channel fails, location device should connect to next available location service.
- **Specification**:

require “position location task” **to** connect **to** next available location service decided by COI **when** (service.Status = “fail”)

Screen Dump: before service G1 fails

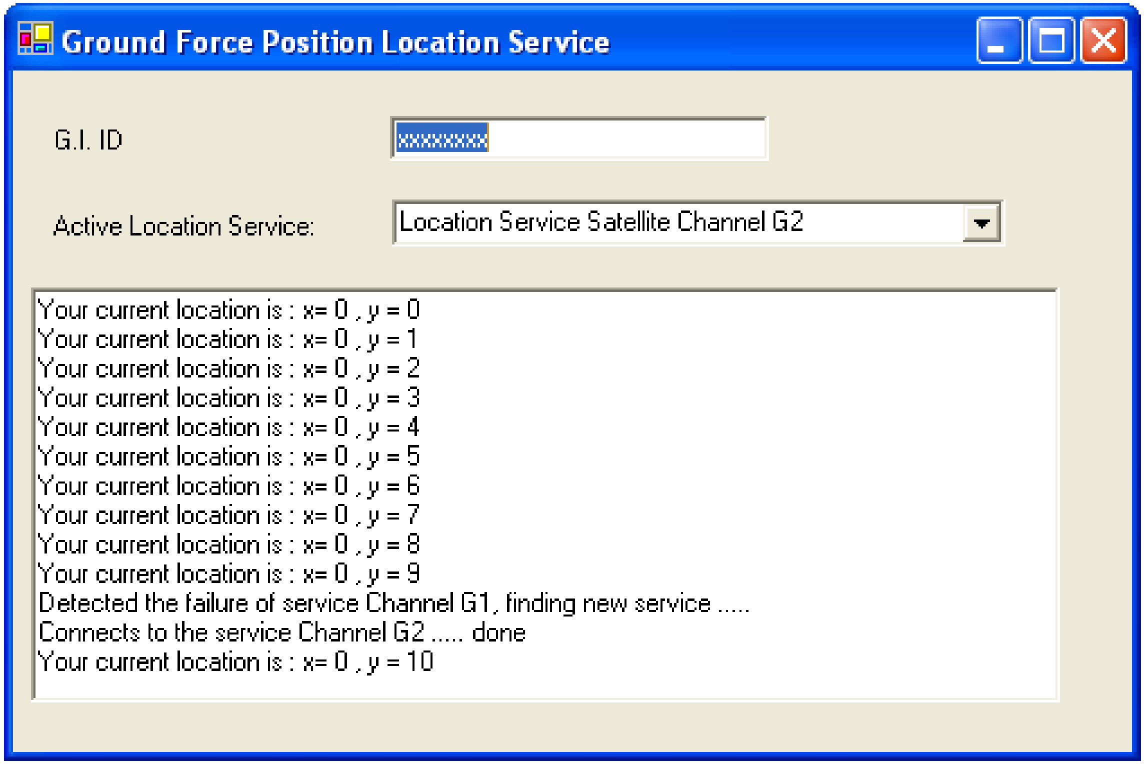


G.I. ID: xxxxxxxxx

Active Location Service: Location Service Satellite Channel G1

```
Your current location is : x= 0 , y = 0
Your current location is : x= 0 , y = 1
Your current location is : x= 0 , y = 2
Your current location is : x= 0 , y = 3
Your current location is : x= 0 , y = 4
Your current location is : x= 0 , y = 5
Your current location is : x= 0 , y = 6
Your current location is : x= 0 , y = 7
Your current location is : x= 0 , y = 8
Your current location is : x= 0 , y = 9
```

Screen Dump: after service G1 fails



G.I. ID: xxxxxxxx

Active Location Service: Location Service Satellite Channel G2

```
Your current location is : x= 0 , y = 0
Your current location is : x= 0 , y = 1
Your current location is : x= 0 , y = 2
Your current location is : x= 0 , y = 3
Your current location is : x= 0 , y = 4
Your current location is : x= 0 , y = 5
Your current location is : x= 0 , y = 6
Your current location is : x= 0 , y = 7
Your current location is : x= 0 , y = 8
Your current location is : x= 0 , y = 9
Detected the failure of service Channel G1, finding new service .....
Connects to the service Channel G2 ..... done
Your current location is : x= 0 , y = 10
```