Agent-Oriented Software Engineering

Nick Jennings

Dept of Electronics and Computer Science

University of Southampton, UK.

nrj@ecs.soton.ac.uk

http://www.ecs.soton.ac.uk/~nrj/

Software Development is Difficult

- **One of most complex construction task humans undertake**

  "Computer science is the first engineering discipline ever in which the complexity of the objects created is limited by the skill of the creator and not limited by the strength of the raw materials. If steel beams were infinitely strong and couldn't ever bend no matter what you did, then skyscrapers could be as complicated as computers." Brian K. Reid

- **True whatever models and techniques are applied**

  "the essential complexity of software" Fred Brooks

- **Software engineering provides models & techniques that make it easier to handle this essential complexity**

Software Development is Getting Harder

- **Shorter development lifecycles**
- **More ambitious requirements**
- **Less certain requirements**
  - Greater scope for change
- **More challenging environments**
  - Greater dynamism
  - Greater openness

Software Engineering: Continually Playing Catch Up

- **Better Models**
  - components
  - design patterns
  - software architectures
  - **interacting agents**
- **Better Processes**
  - light methods
  - heavier methods

"Our ability to imagine complex applications will always exceed our ability to develop them"

Grady Booch

The Adequacy Hypothesis

Agent-oriented approaches can enhance our ability to model, design and build complex distributed software systems.

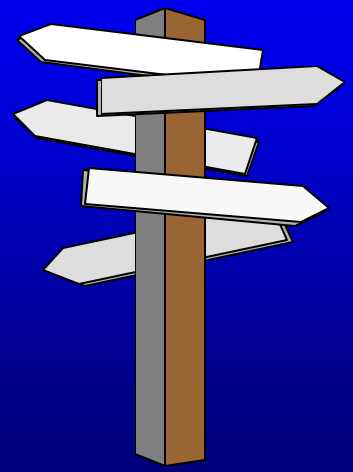
Talk Outline

I. The Essence of Agent-Based Computing

II. The Case for Agent-Oriented Software Engineering

III. Potential Drawbacks

IV. Conclusions



Talk Outline

I. The Essence of Agent-Based Computing

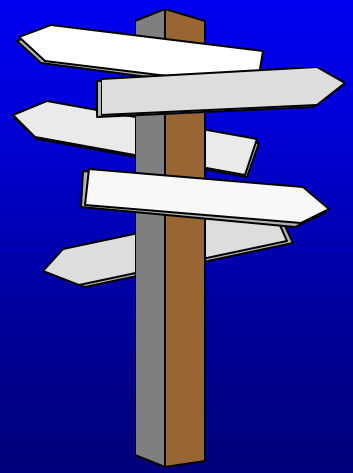
II. The Case for Agent-Oriented Software Engineering

III. Potential Drawbacks

IV. Conclusions

"The intolerable wrestle with words and meanings."

T. S. Eliot



Agent

“encapsulated computer system, situated in some environment, and capable of flexible autonomous action in that environment in order to meet its design objectives” (Wooldridge)

Agent

"encapsulated computer system, situated in some environment, and capable of flexible autonomous action in that environment in order to meet its design objectives" (Wooldridge)

- control over internal state and over own behaviour

Agent

"encapsulated computer system, situated in some environment, and capable of flexible autonomous action in that environment in order to meet its design objectives" (Wooldridge)

- control over internal state and over own behaviour
- experiences environment through sensors and acts through effectors

Agent

"encapsulated computer system, situated in some environment, and capable of **flexible** autonomous action in that environment in order to meet its design objectives" (Wooldridge)

- control over internal state and over own behaviour
- experiences environment through sensors and acts through effectors

- reactive: respond in timely fashion to environmental change
- proactive: act in anticipation of future goals

Definitional Malaise

“My guess is that object-oriented programming will be what structured programming was in the 1970s. Everybody will be in favour of it. Every manufacturer will promote his product as supporting it. Every manager will pay lip service to it. Every programmer will practice it (differently). And no one will know just what it is.” (Rentsch, 82)

“My guess is that agent-based computing will be what object-oriented programming was in the 1980s. Everybody will be in favour of it. Every manufacturer will promote his product as supporting it. Every manager will pay lip service to it. Every programmer will practice it (differently). And no one will know just what it is.” (Jennings, 00)

Multiple Agents

In most cases, single agent is insufficient

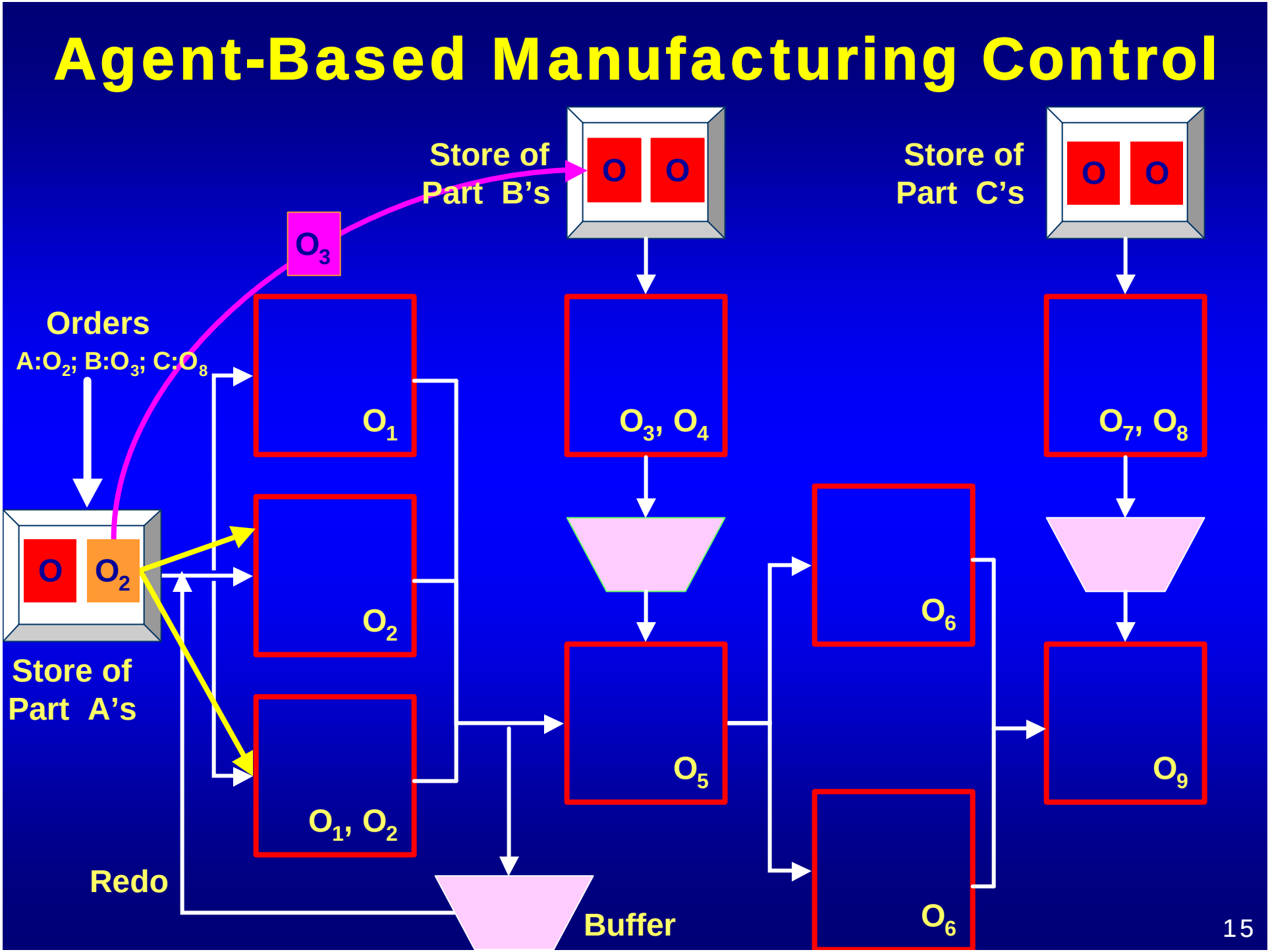
- no such thing as a single agent system (!?)
- multiple agents are the norm, to represent:
  - **natural decentralisation**
  - **multiple loci of control**
  - **multiple perspectives**
  - **competing interests**

Agent Interactions

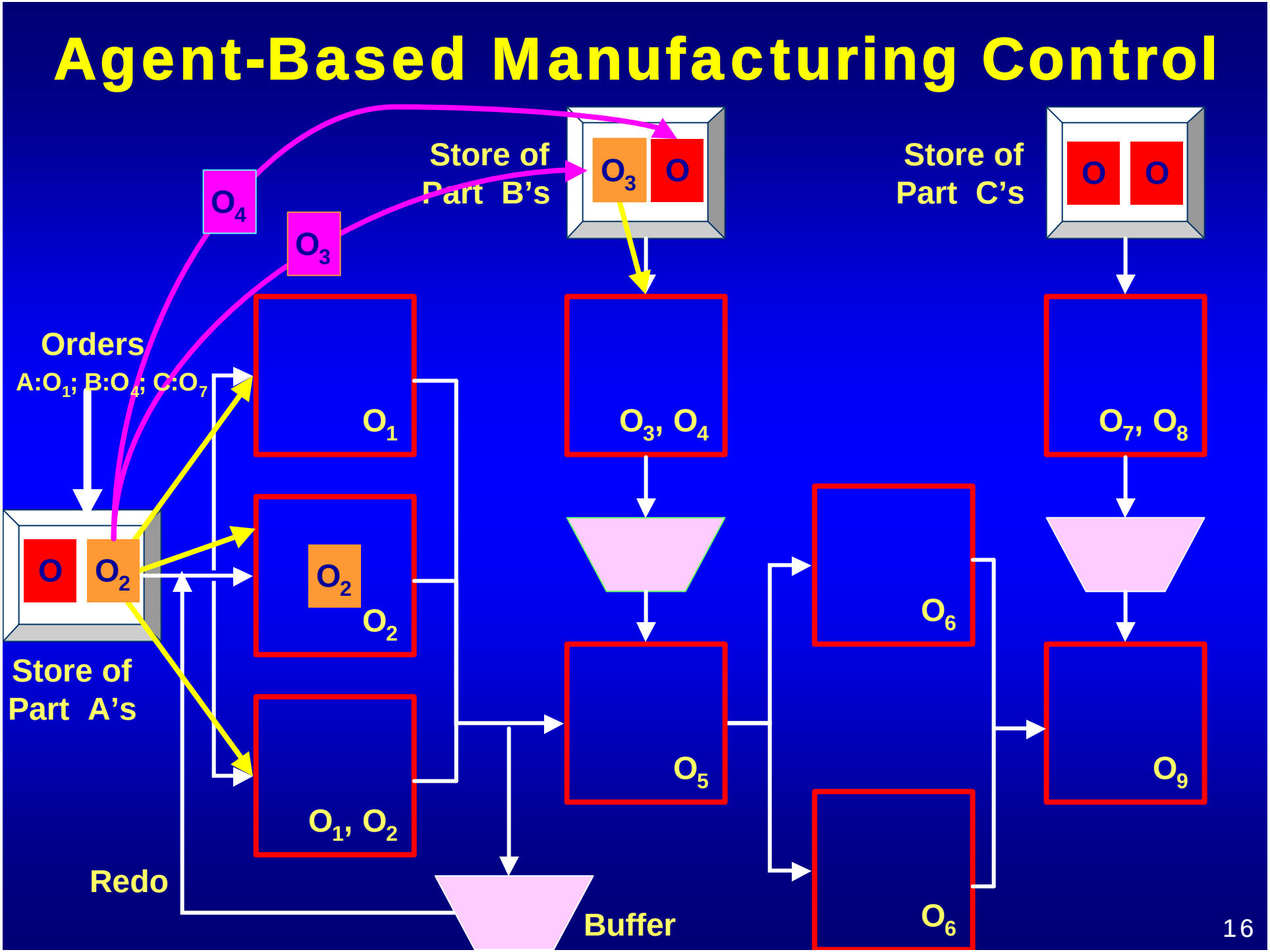
- **Interaction between agents is inevitable**
  - to achieve individual objectives, to manage inter-dependencies
- **Conceptualised as taking place at knowledge-level**
  - which goals, at what time, by whom, what for
- **Flexible run-time initiation and responses**
  - cf. design-time, hard-wired nature of extant approaches

paradigm shift from previous perceptions of computational interaction

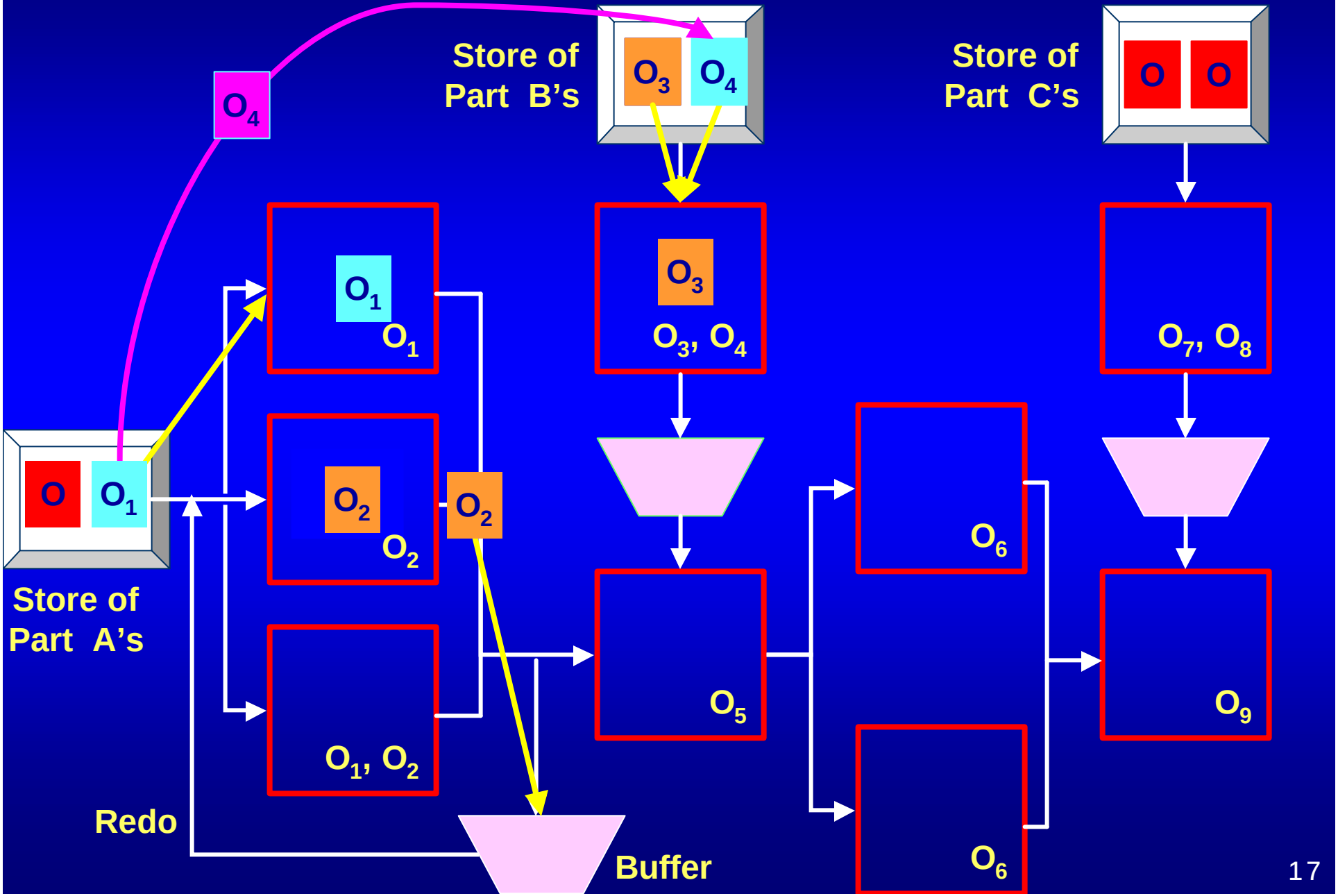
Agent-Based Manufacturing Control



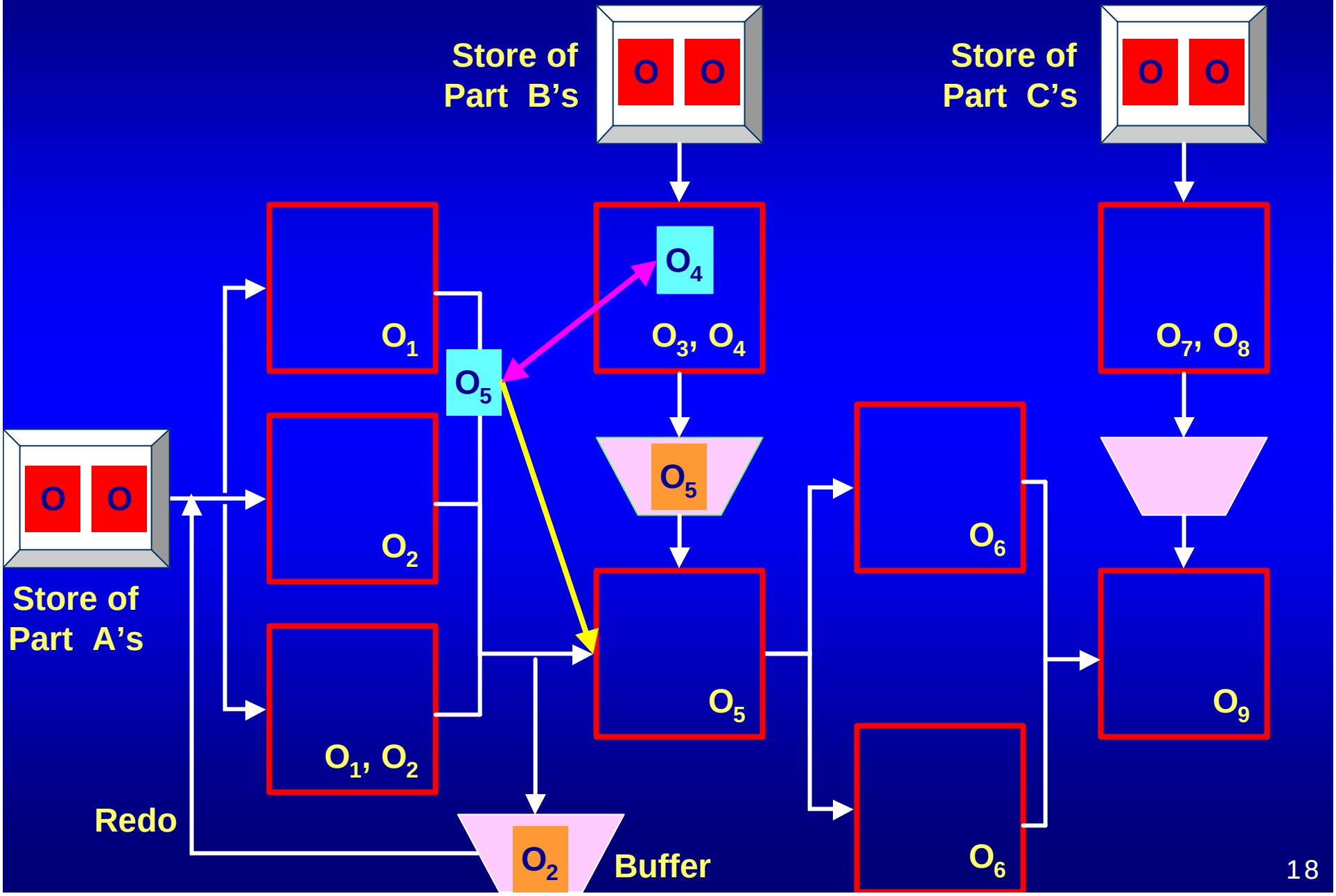
Agent-Based Manufacturing Control



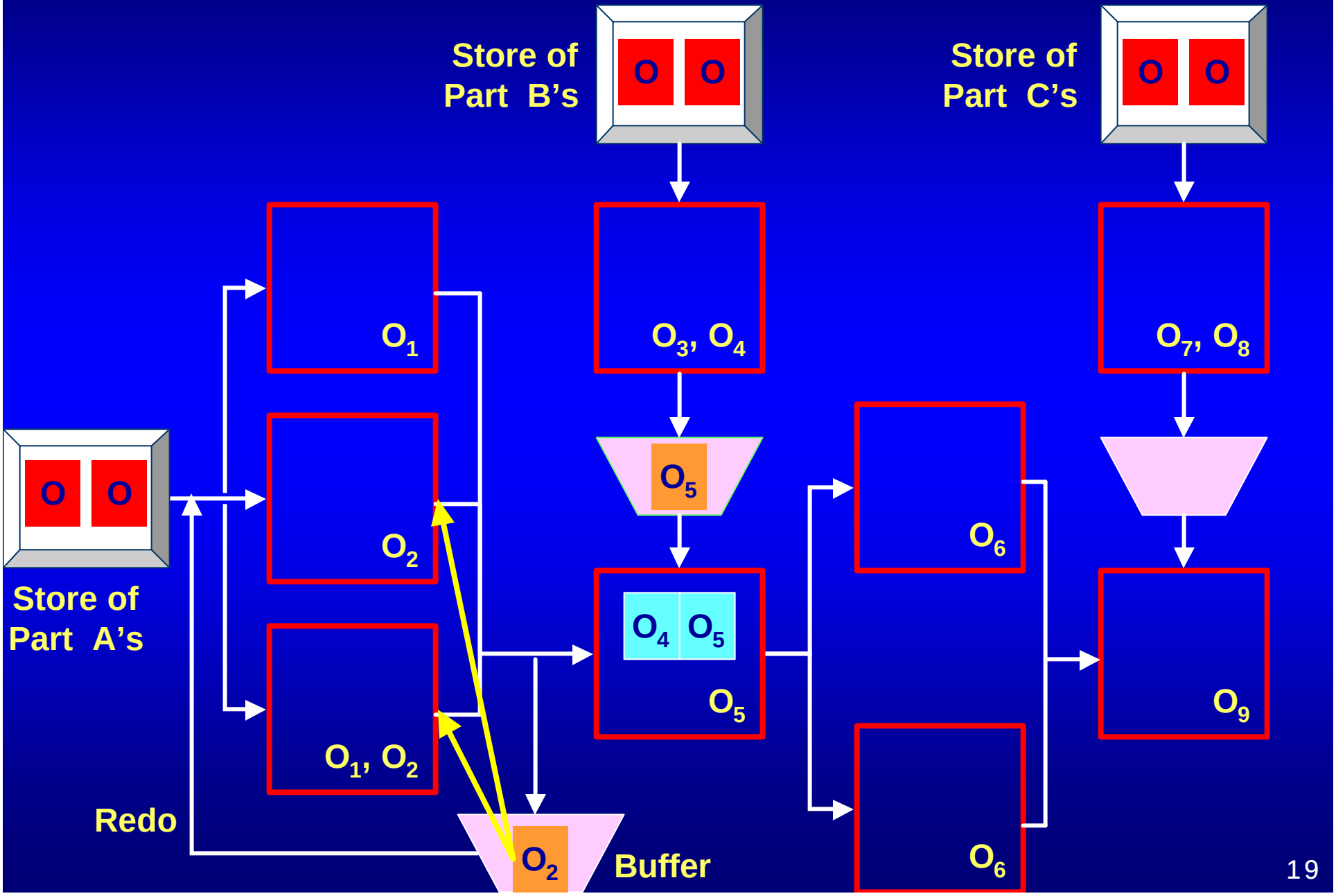
Agent-Based Manufacturing Control



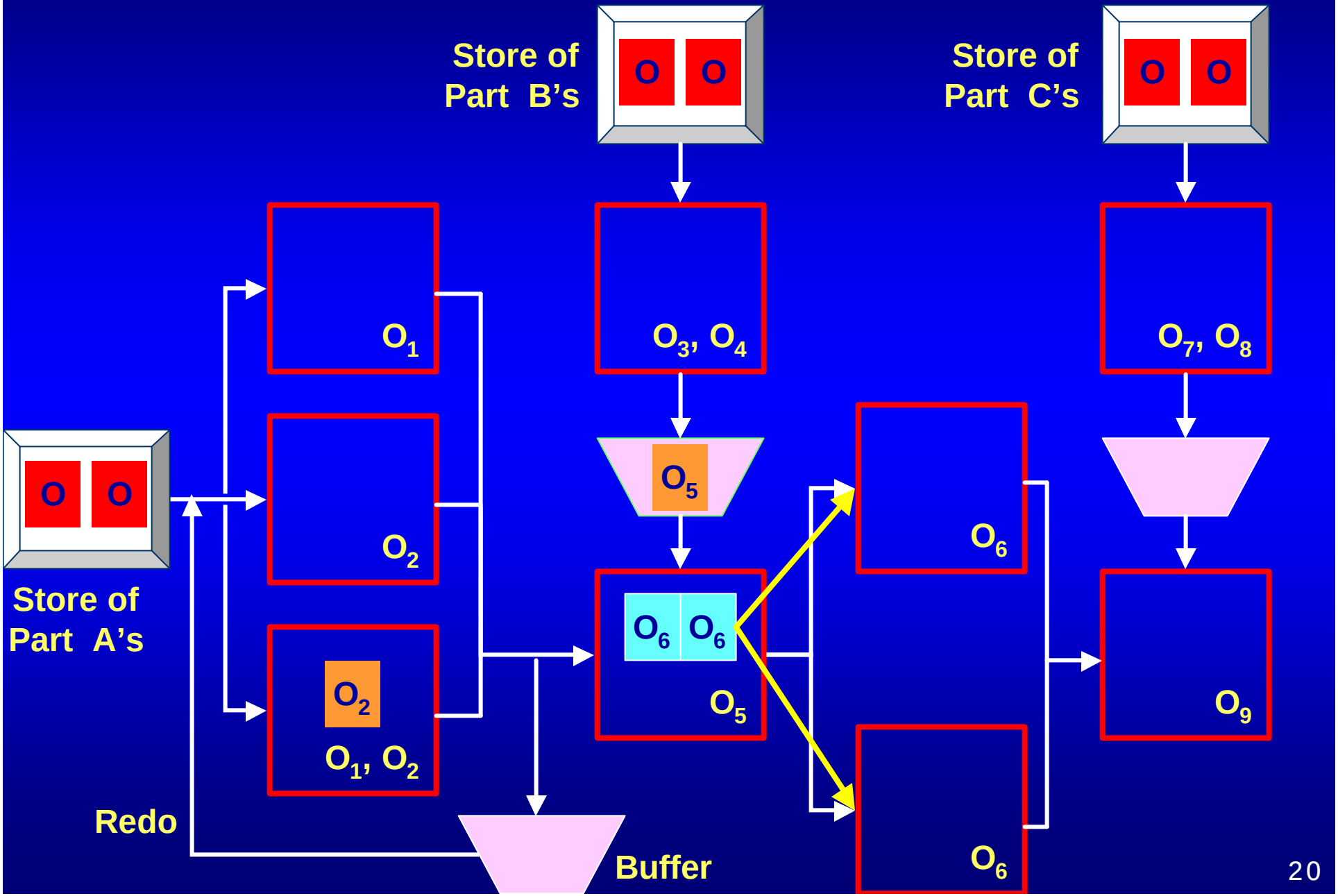
Agent-Based Manufacturing Control



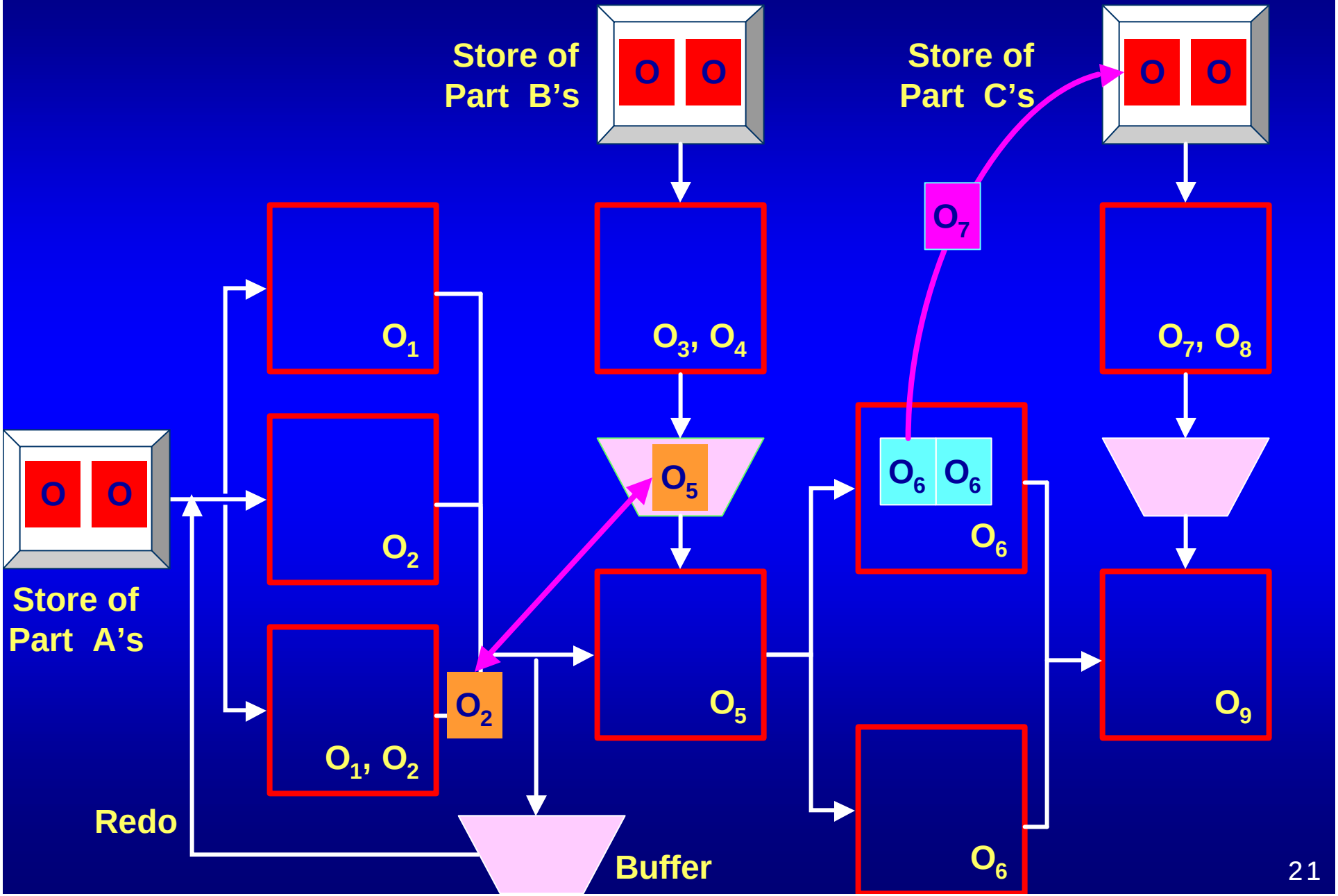
Agent-Based Manufacturing Control



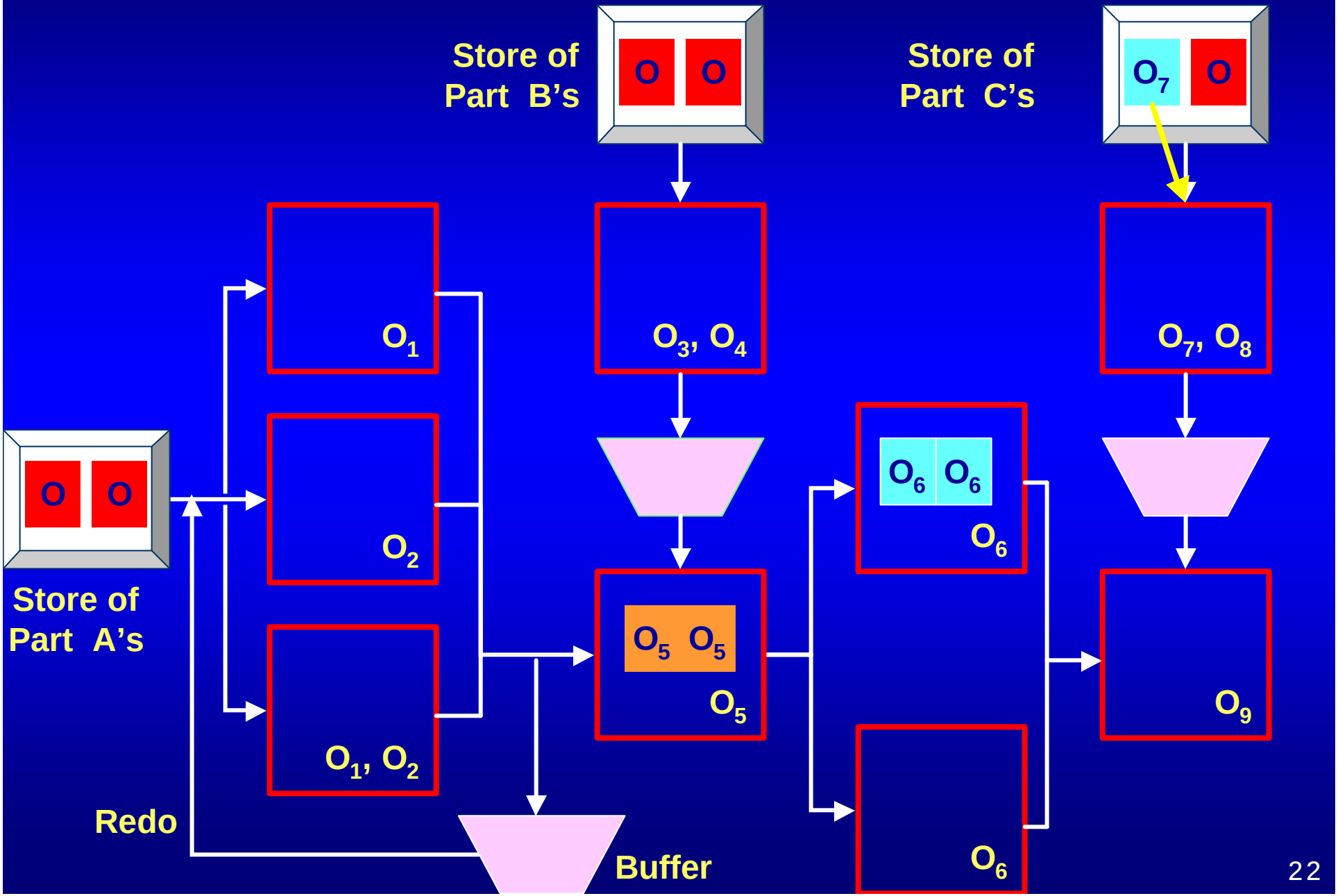
Agent-Based Manufacturing Control



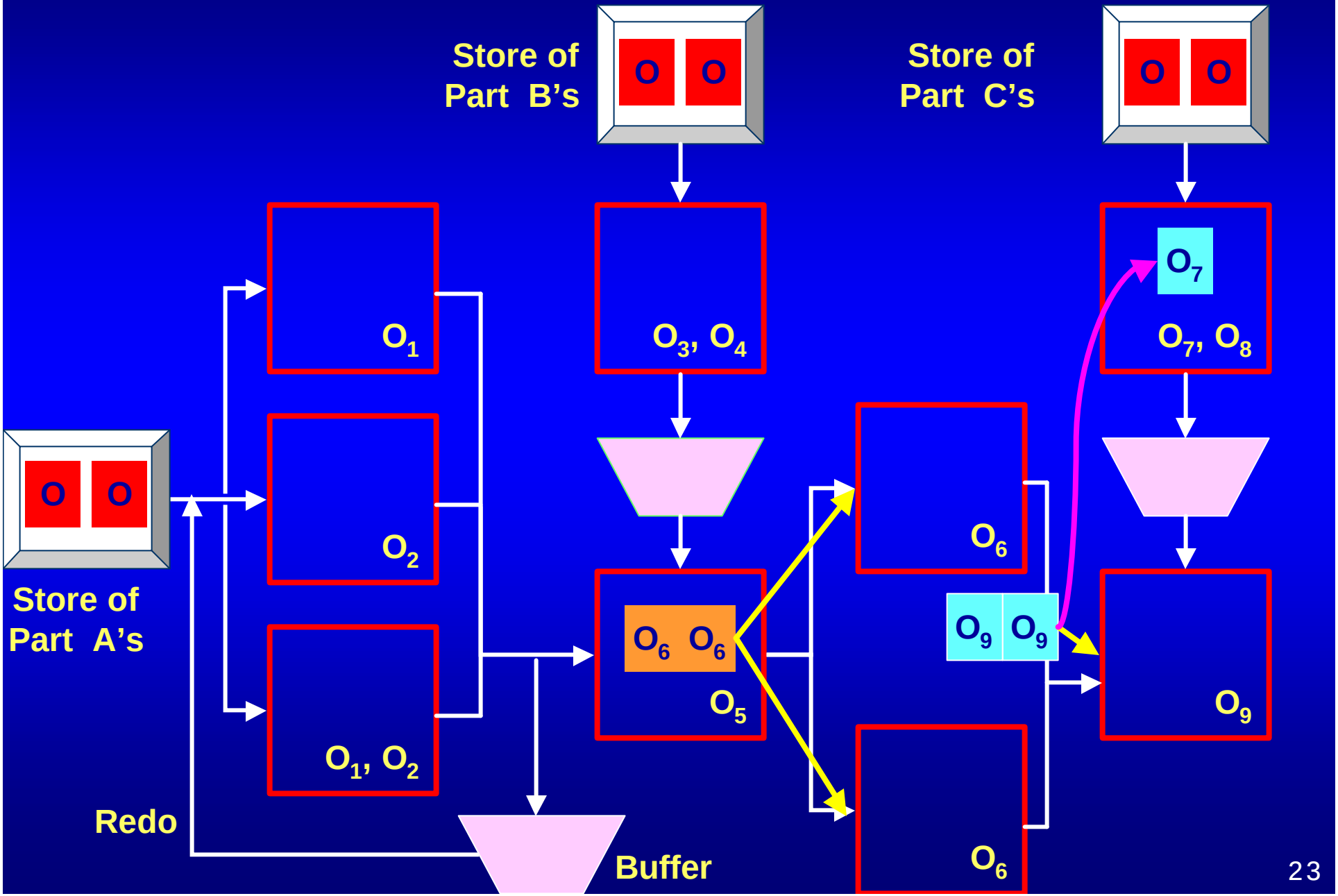
Agent-Based Manufacturing Control



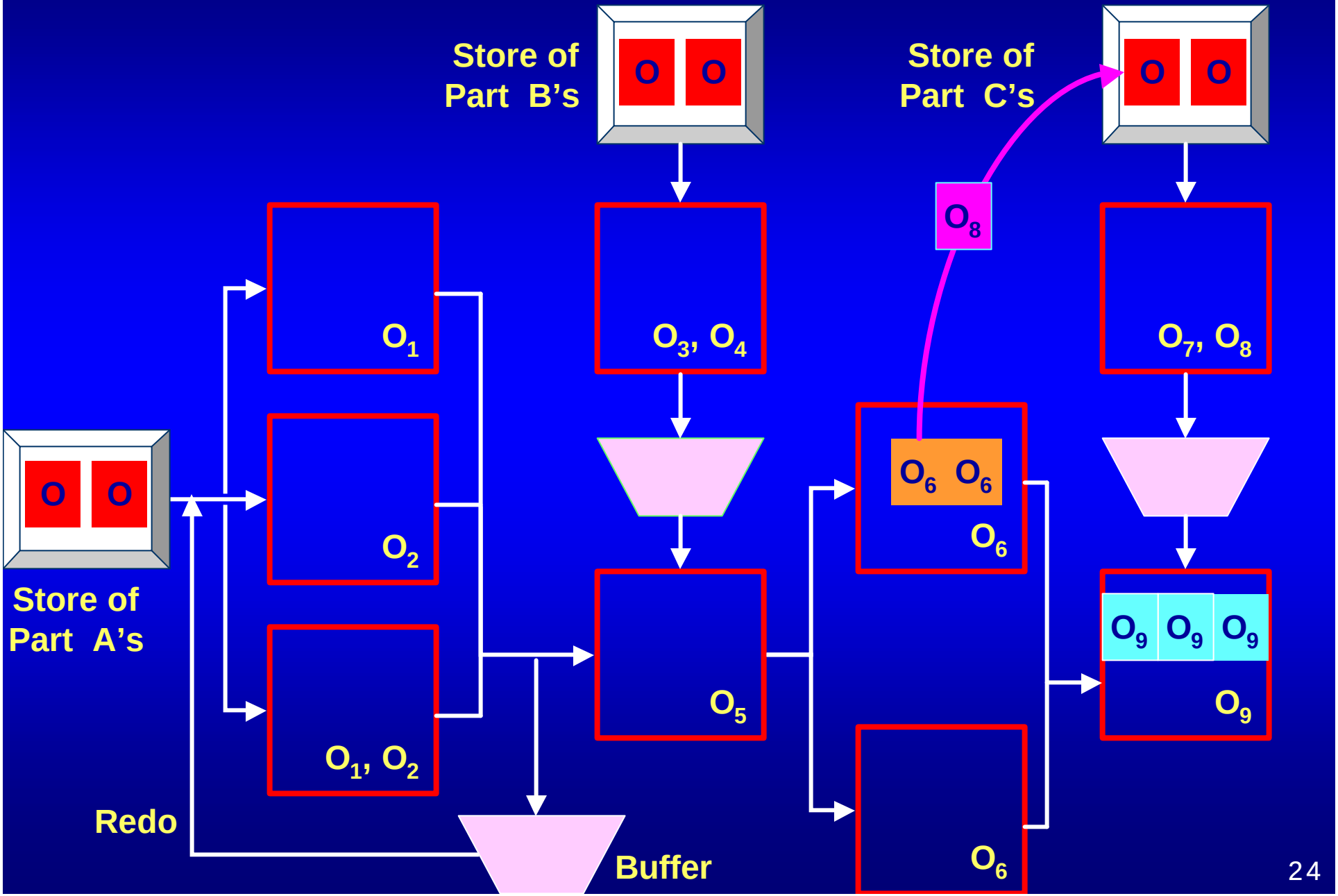
Agent-Based Manufacturing Control



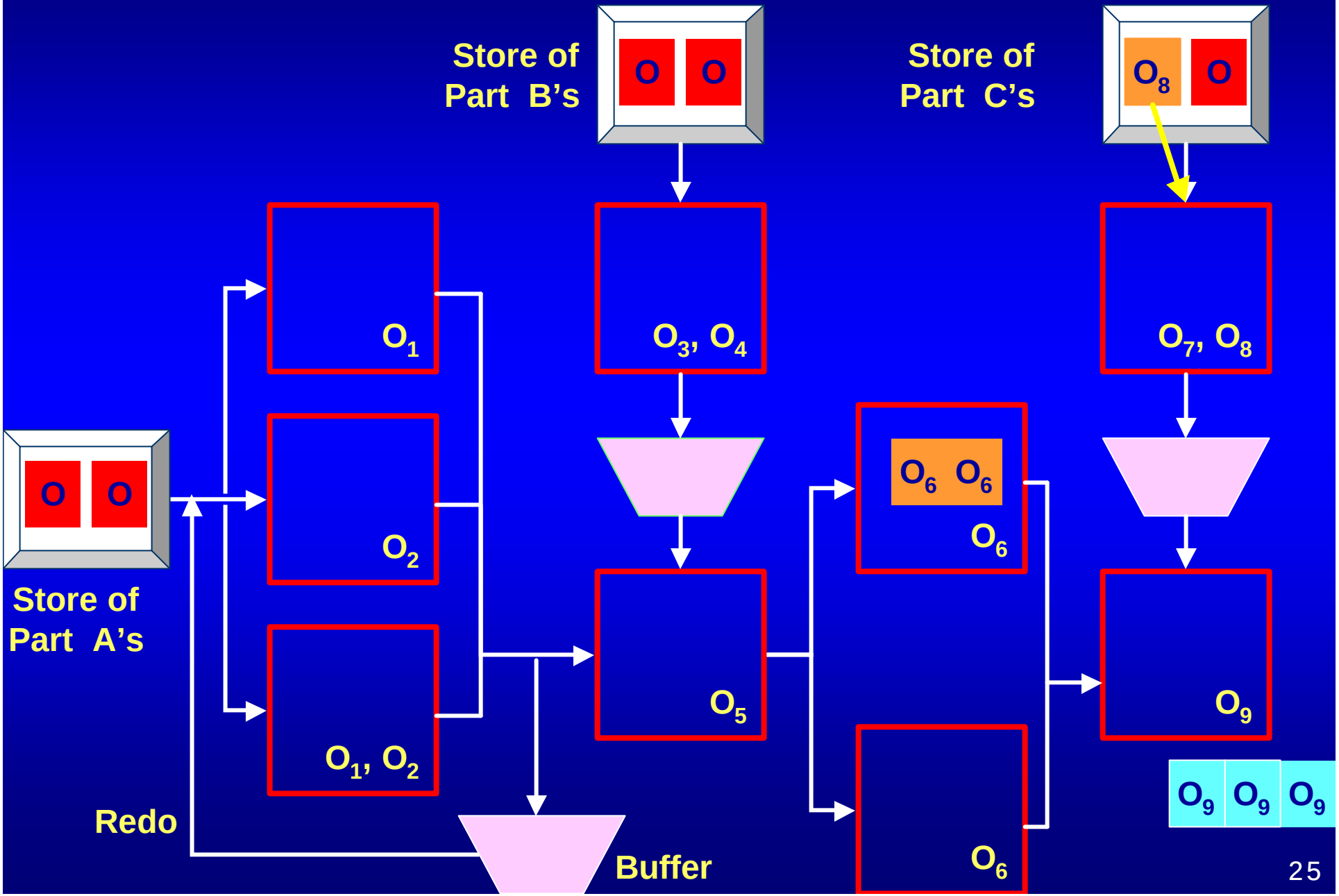
Agent-Based Manufacturing Control



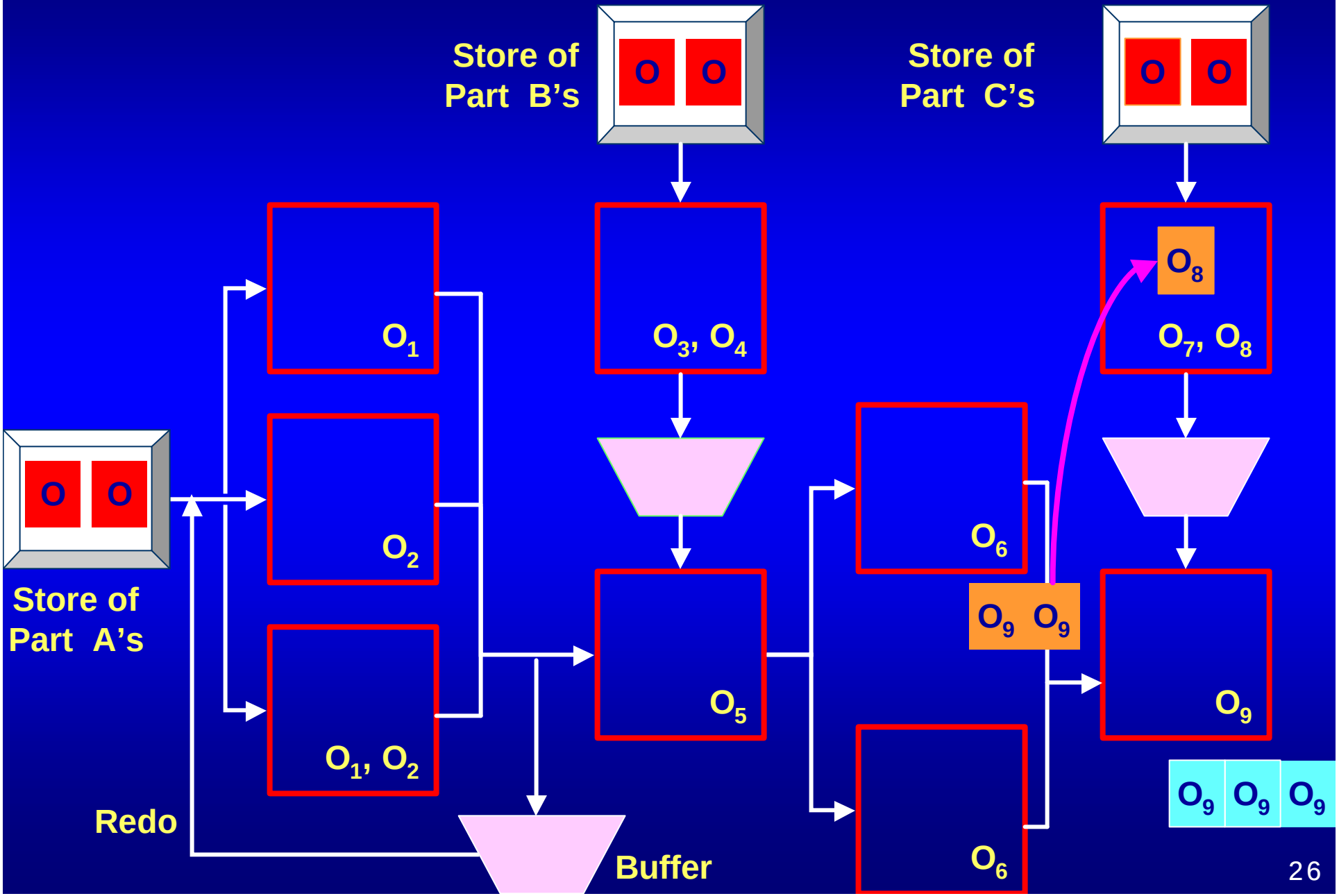
Agent-Based Manufacturing Control



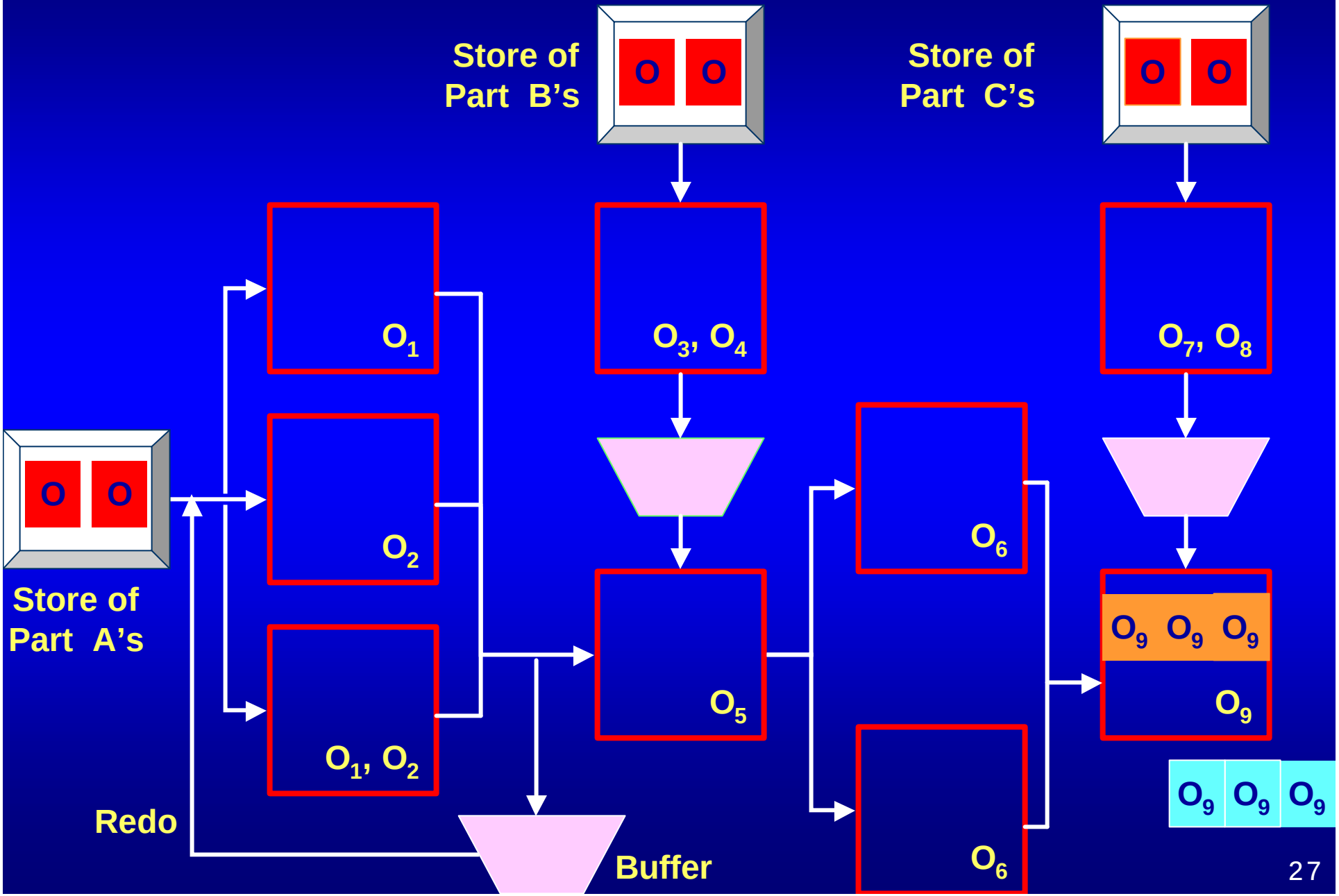
Agent-Based Manufacturing Control



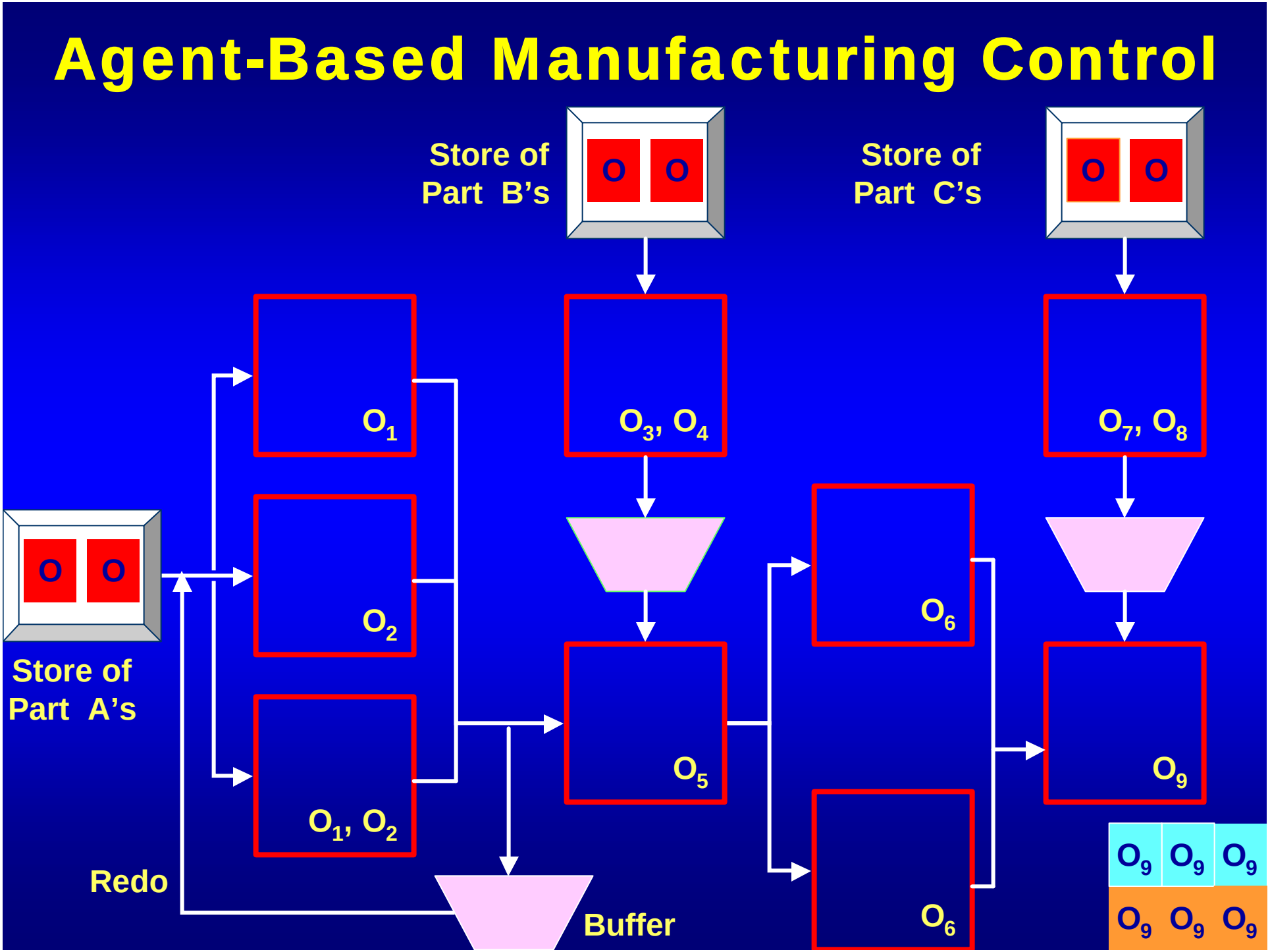
Agent-Based Manufacturing Control



Agent-Based Manufacturing Control



Agent-Based Manufacturing Control

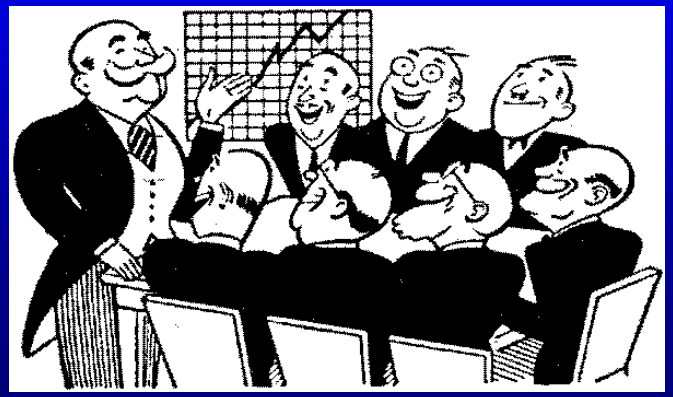


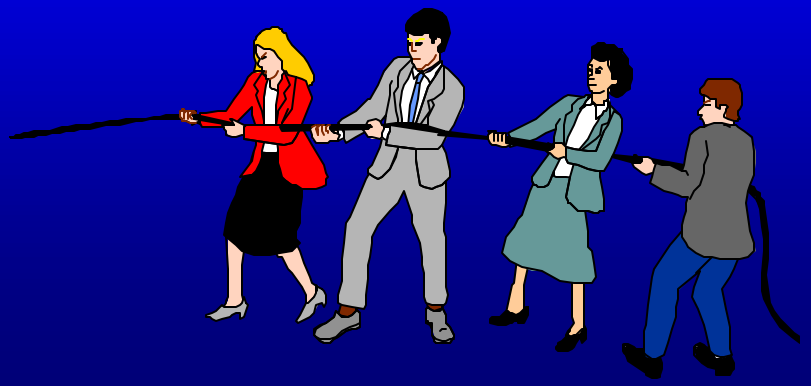
Organisations

Agents act/interact to achieve objectives:

- on behalf of individuals/companies
- part of a wider problem solving initiative

→ underlying organisational relationship between the agents



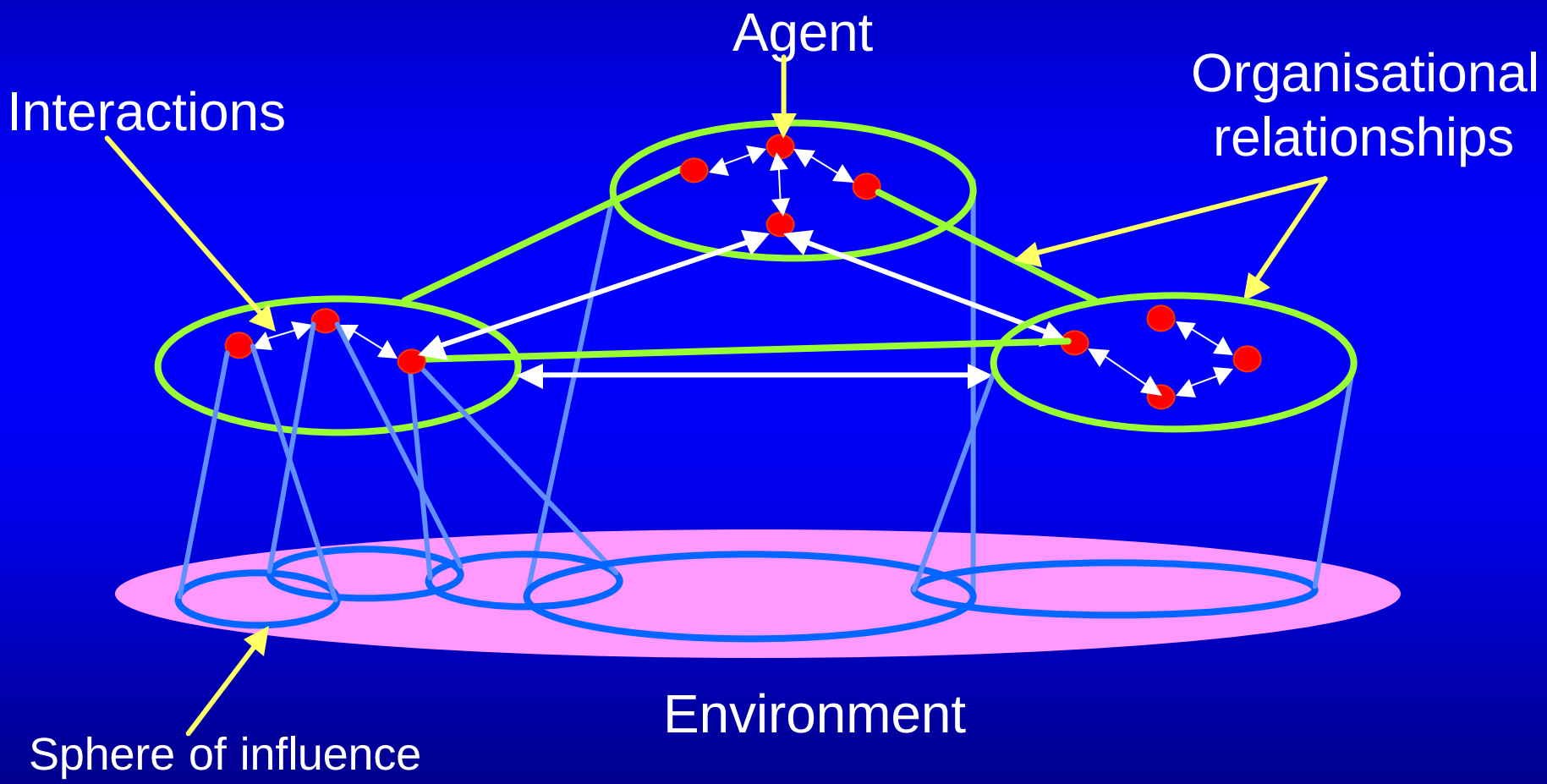


Organisations

This organisational context:

- influences agents' behaviour
  - **relationships need to be made explicit**
    - peers
    - teams, coalitions
    - authority relationships
- is subject to ongoing change
  - **provide computational apparatus for creating, maintaining and disbanding structures**

A Canonical View



(see also: Castelfranchi, Ferber, Gasser, Lesser, …..)

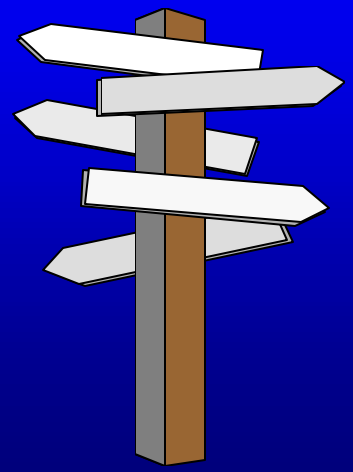
Talk Outline

I. The Essence of Agent-Based Computing

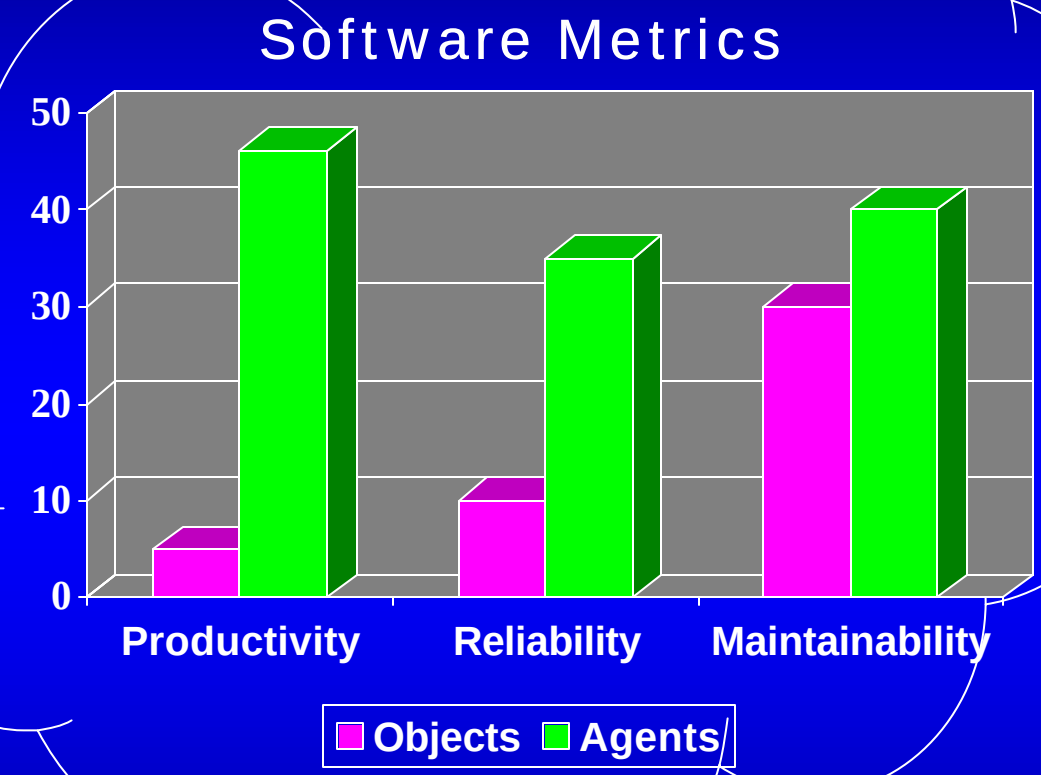
II. The Case for Agent-Oriented Software Engineering

III. Potential Drawbacks

IV. Conclusions

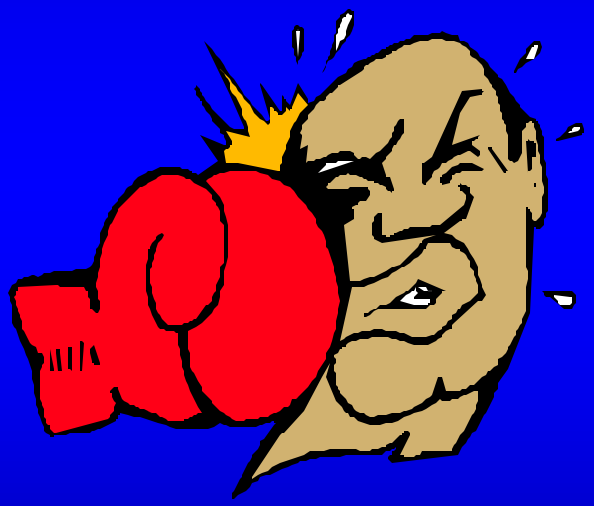


Making the Case: Quantitatively



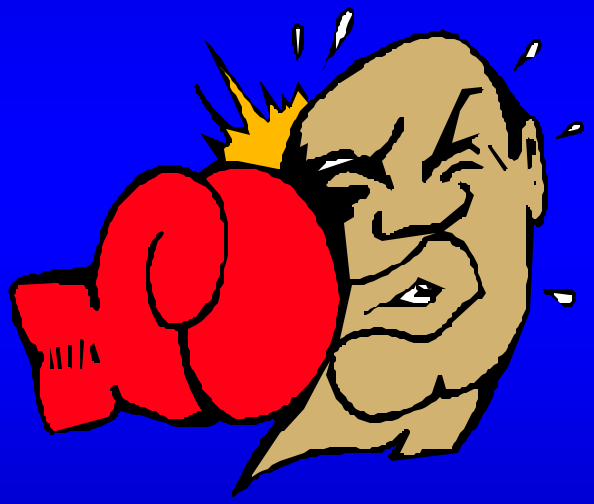
"There are 3 kinds of lies: lies, damned lies and statistics"
Disraeli

Making the Case: Qualitatively



Making the Case: Qualitatively

Software techniques for
tackling complexity

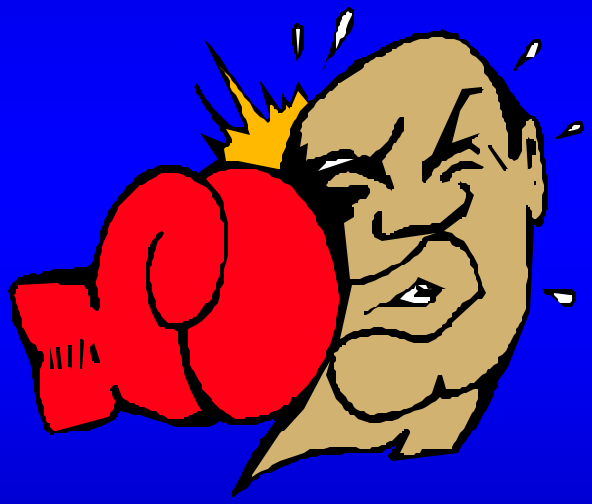


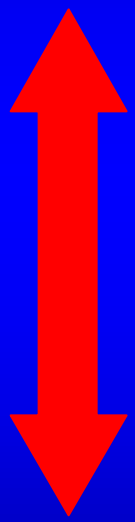
Tackling Complexity

- Decomposition
- Abstraction
- Organisation

Making the Case: Qualitatively

Software techniques for tackling complexity





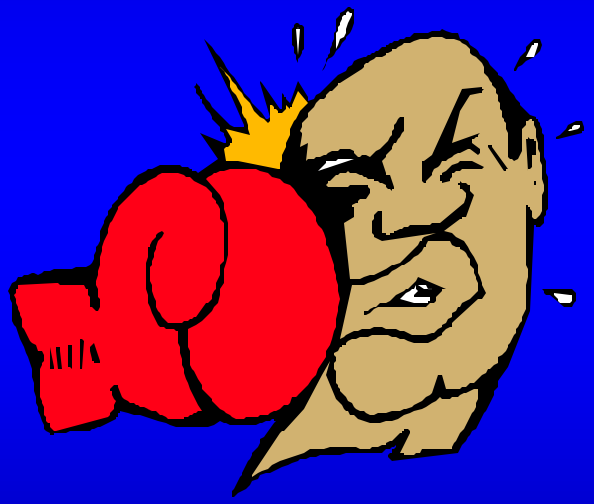
Nature of complex systems

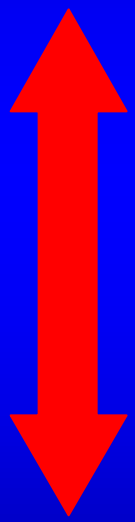
Complex Systems (Herb Simon)

- **Complexity takes form of "hierarchy"**
  - not a control hierarchy
  - collection of related sub-systems at different levels of abstraction
- **Can distinguish between interactions among sub-systems and interactions within sub-systems**
  - latter more frequent & predictable: "nearly decomposable systems"
- **Arbitrary choice about which components are primitive**
- **Systems that support evolutionary growth develop more quickly than those that do not: "stable intermediate forms"**

Making the Case: Qualitatively

Software techniques for tackling complexity



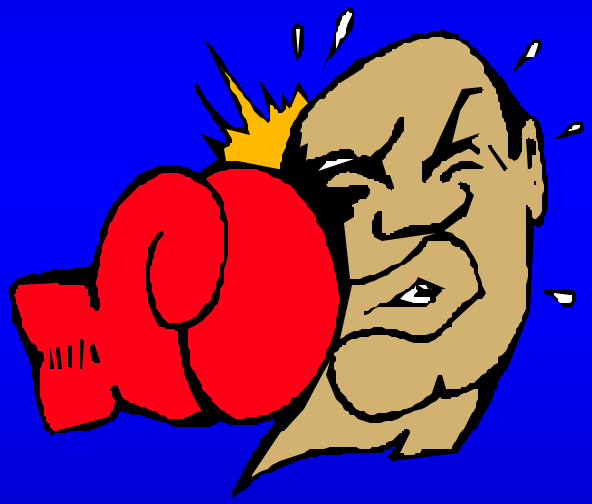


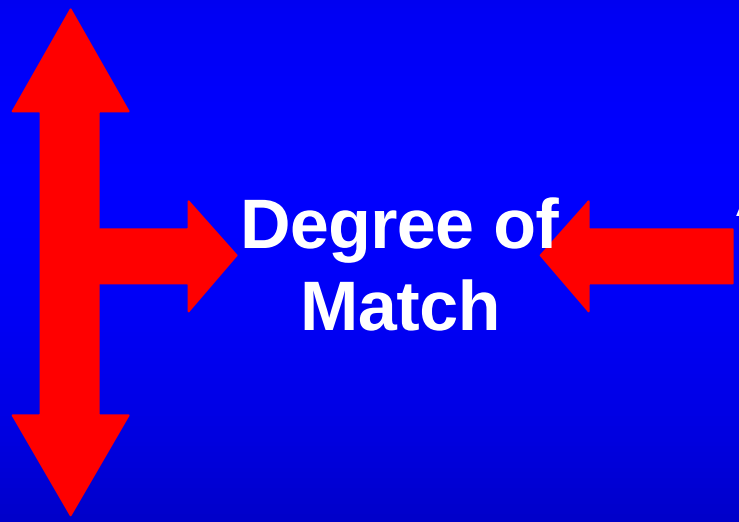
Nature of complex systems

Agent-based computing

Making the Case: Qualitatively

Software techniques for tackling complexity





Degree of Match

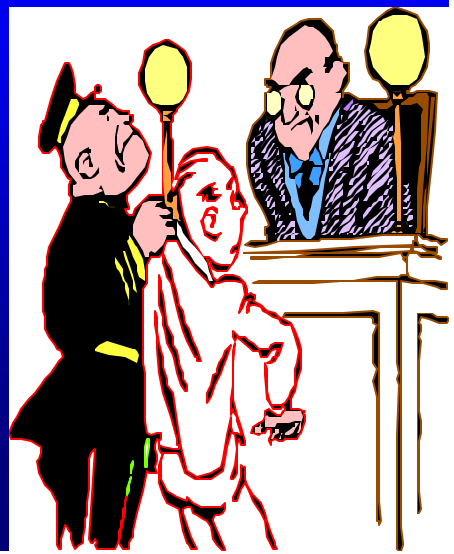
Agent-based computing

Nature of complex systems

The Match Process

1. Show agent-oriented decomposition is effective way of partitioning problem space of complex system

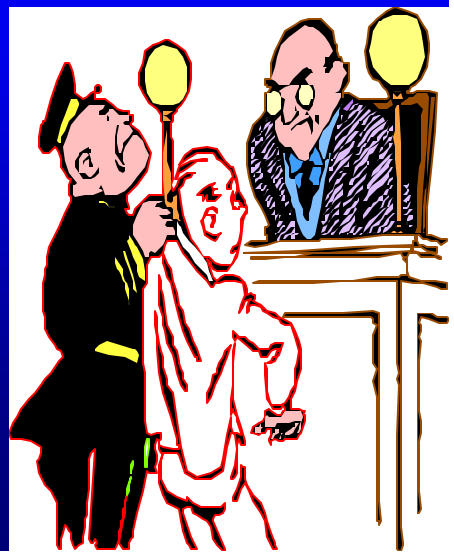
2. Show key abstractions of agent-oriented mindset are natural means of modelling complex systems



The Match Process

1. Show agent-oriented decomposition is effective way of partitioning problem space of complex system

2. Show key abstractions of agent-oriented mindset are natural means of modelling complex systems



Decomposition: Agents

- **In terms of entities that have:**
  - own persistent thread of control (active: “say go”)
  - control over their own destiny (autonomous: “say no”)
- **Makes engineering of complex systems easier:**
  - natural representation of multiple loci of control
    - **“real systems have no top” (Meyer)**
  - allows competing objectives to be represented and reconciled in context sensitive fashion

Decomposition: Interactions

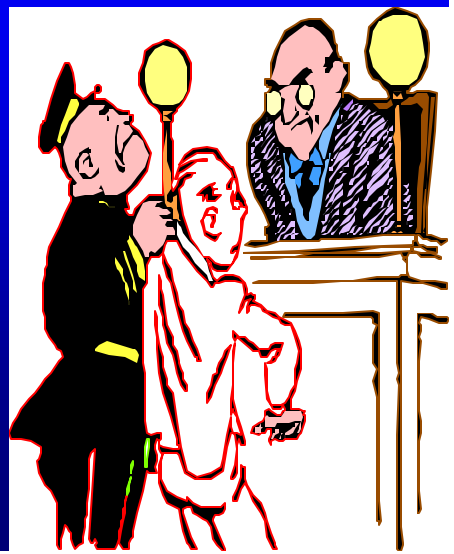
- **Agents make decisions about nature & scope of interactions at run time**

- **Makes engineering of complex systems easier:**
  - unexpected interaction is expected
    - **not all interactions need be set at design time**
  - simplified management of control relationships between components
    - **coordination occurs on as-needed basis between continuously active entities**

The Match Process

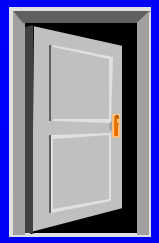
1. Show agent-oriented decomposition is effective way of partitioning problem space of complex system ✓

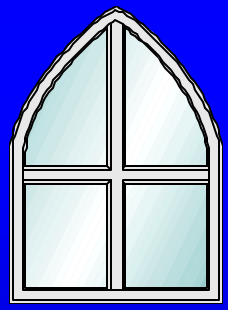
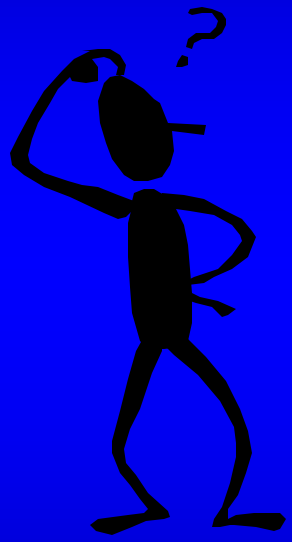
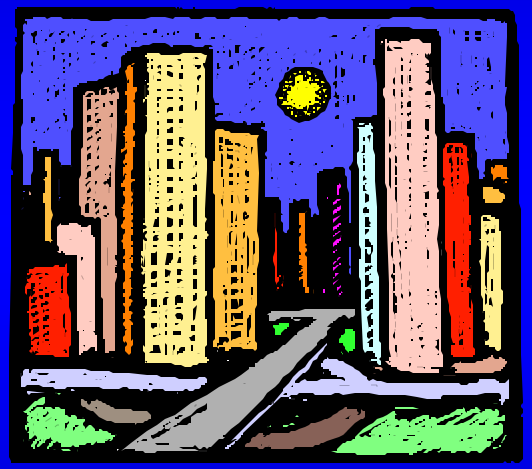
2. Show key abstractions of agent-oriented mindset are natural means of modelling complex systems



Suitability of Abstractions

- **Design is about having right models**



- **In software, minimise gap between units of analysis and constructs of solution paradigm**
  - OO techniques natural way of modelling world

Complex System	Agent-Based System
Sub-systems	
Sub-system components	
Interactions between sub-systems and sub-system components	
Relationships between sub-systems and sub-system components	

Complex System	Agent-Based System
Sub-systems	Agent organisations
Sub-system components	
Interactions between sub-systems and sub-system components	
Relationships between sub-systems and sub-system components	

Complex System	Agent-Based System
Sub-systems	Agent organisations
Sub-system components	Agents
Interactions between sub-systems and sub-system components	
Relationships between sub-systems and sub-system components	

Complex System	Agent-Based System
Sub-systems	Agent organisations
Sub-system components	Agents
Interactions between sub-systems and sub-system components: "at any given level of abstraction, find meaningful collections of entities that collaborate to achieve some higher level view" (Booch)	"cooperating to achieve common objectives" "coordinating their actions" "negotiating to resolve conflicts"
Relationships between sub-systems and sub-system components	

Complex System	Agent-Based System
Sub-systems	Agent organisations
Sub-system components	Agents
Interactions between sub-systems and sub-system components	“cooperating to achieve common objectives” “coordinating their actions” “negotiating to resolve conflicts”
Relationships between sub-systems and sub-system components - change over time - treat collections as single coherent unit	Explicit mechanisms for representing & managing organisational relationships Structures for modelling collectives

The Adequacy Hypothesis

Agent-oriented approaches can enhance our ability to model, design and build complex distributed software systems.

The Establishment Hypothesis

As well as being suitable for designing and building complex systems,

agents will succeed as a software engineering paradigm

[NB: will be complementary to existing software models like OO, patterns, components, …]

Agents Consistent with Trends in Software Engineering

- **Conceptual basis rooted in problem domain**
  - world contains autonomous entities that interact to get things done

Agents Consistent with Trends in Software Engineering

- **Conceptual basis rooted in problem domain**
- **Increasing localisation and encapsulation**
  - apply to control, as well as state and behaviour

Agents Consistent with Trends in Software Engineering

- **Conceptual basis rooted in problem domain**
- **Increasing localisation and encapsulation**
- **Greater support for re-use of designs and programs**
  - whole sub-system components (cf. components, patterns)
    - **e.g. agent architectures, system structures**
  - flexible interactions (cf. patterns, architectures)
    - **e.g. contract net protocol, auction protocols**

Agents Support System Development by Synthesis

An agent is a stable intermediate form

- able to operate to achieve its objectives and interact with others in flexible ways

  ➡ **construct "system" by bringing agents together and watching overall functionality emerge from their interplay**

- well suited to developments in:
  - **open systems (e.g. Internet)**
  - **e-commerce**

Talk Outline

I. The Essence of Agent-Based Computing

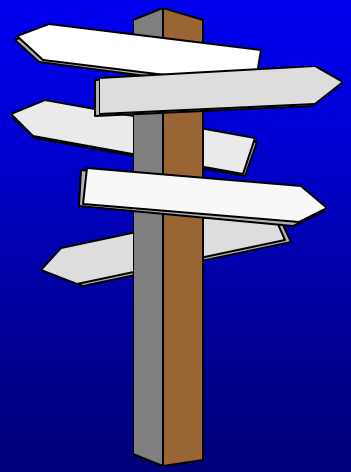
II. The Case for Agent-Oriented Software Engineering

III. Potential Drawbacks

IV. Conclusions

"The moment we want to believe something, we suddenly see all the arguments for it, and become blind to the arguments against it"

George Bernard Shaw

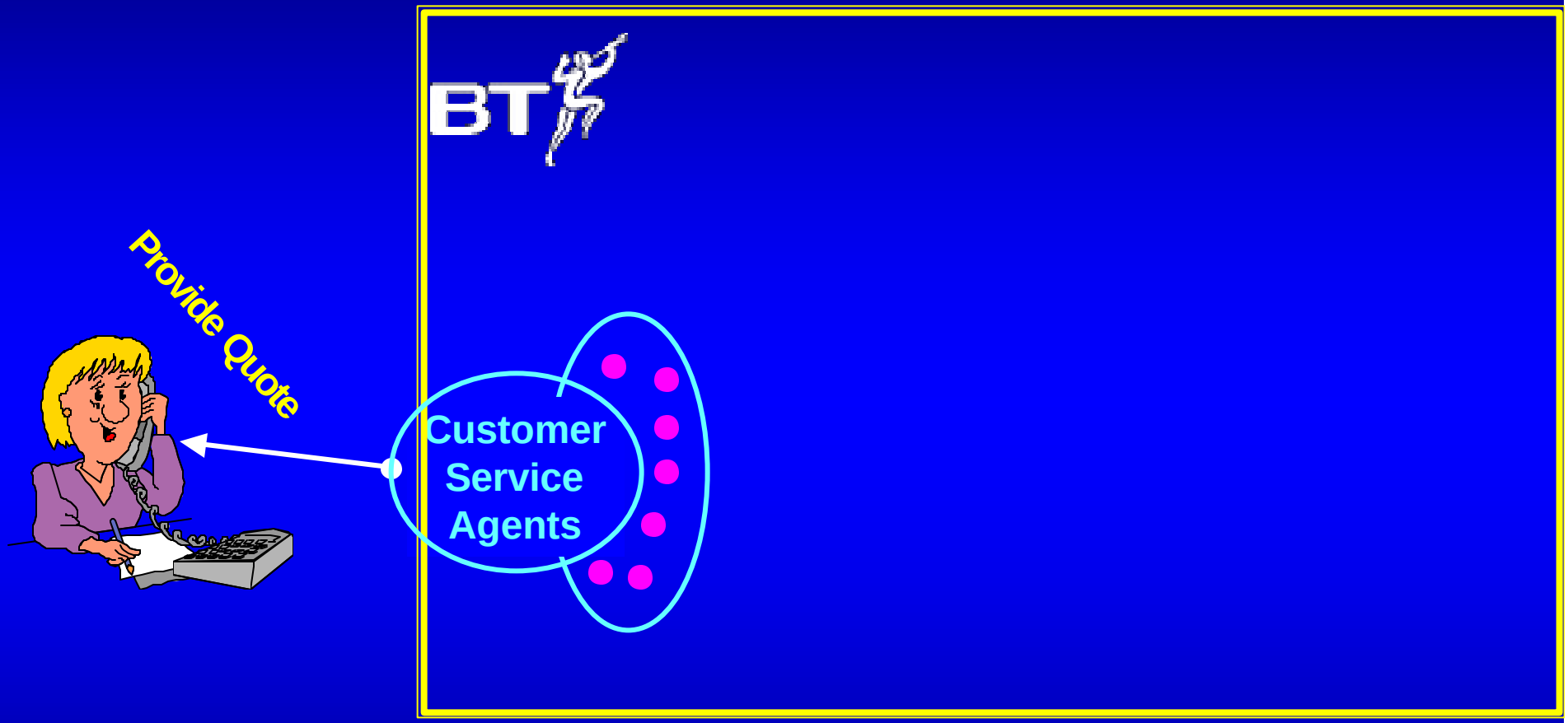


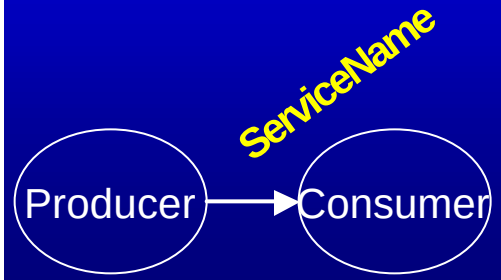
Isn’t autonomous software and making decisions about interactions at run-time a recipe for disaster?

- Yes
  - If expect to just throw agents together and have a coherent and efficient system.
    - **Sometimes this is exactly what is desired**
      - Simulation software
    - **Also the default for synthesised systems**
- No
  - If engineer system appropriately
    - **Interaction Engineering**
    - **Organisation Engineering**

BT's Provide Customer Quote Process

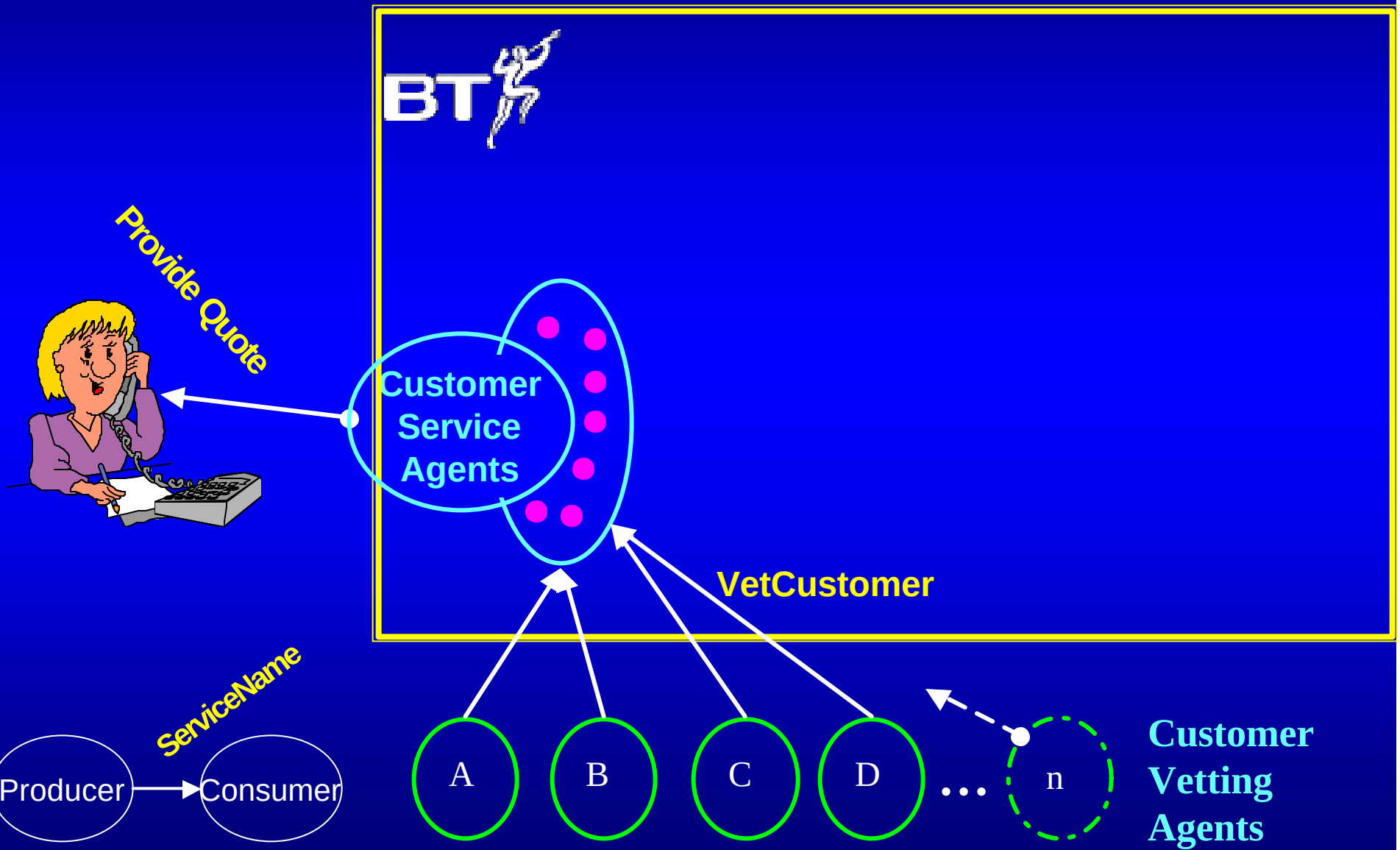
(Jennings *et al.)*





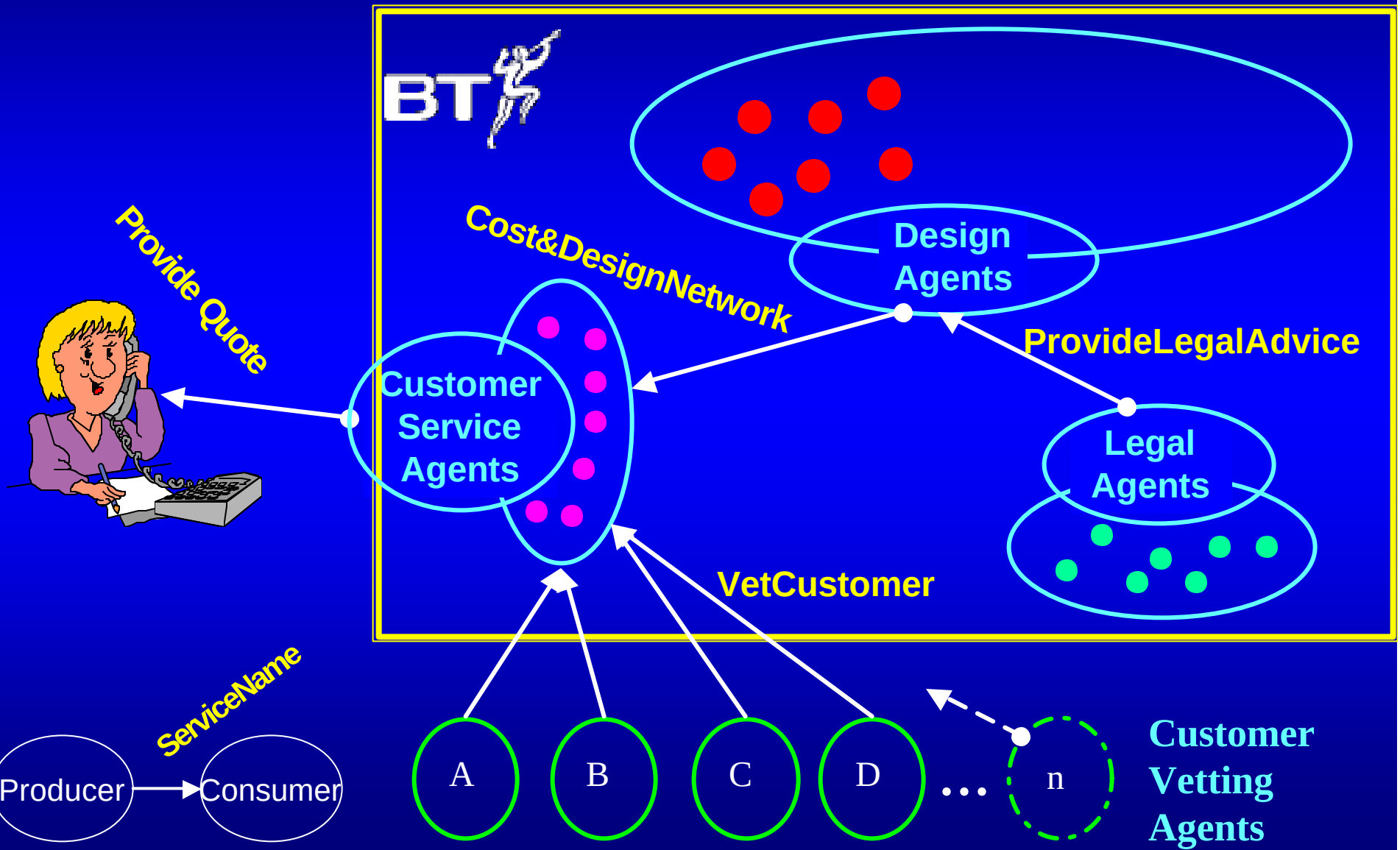
BT's Provide Customer Quote Process

(Jennings *et al.*)



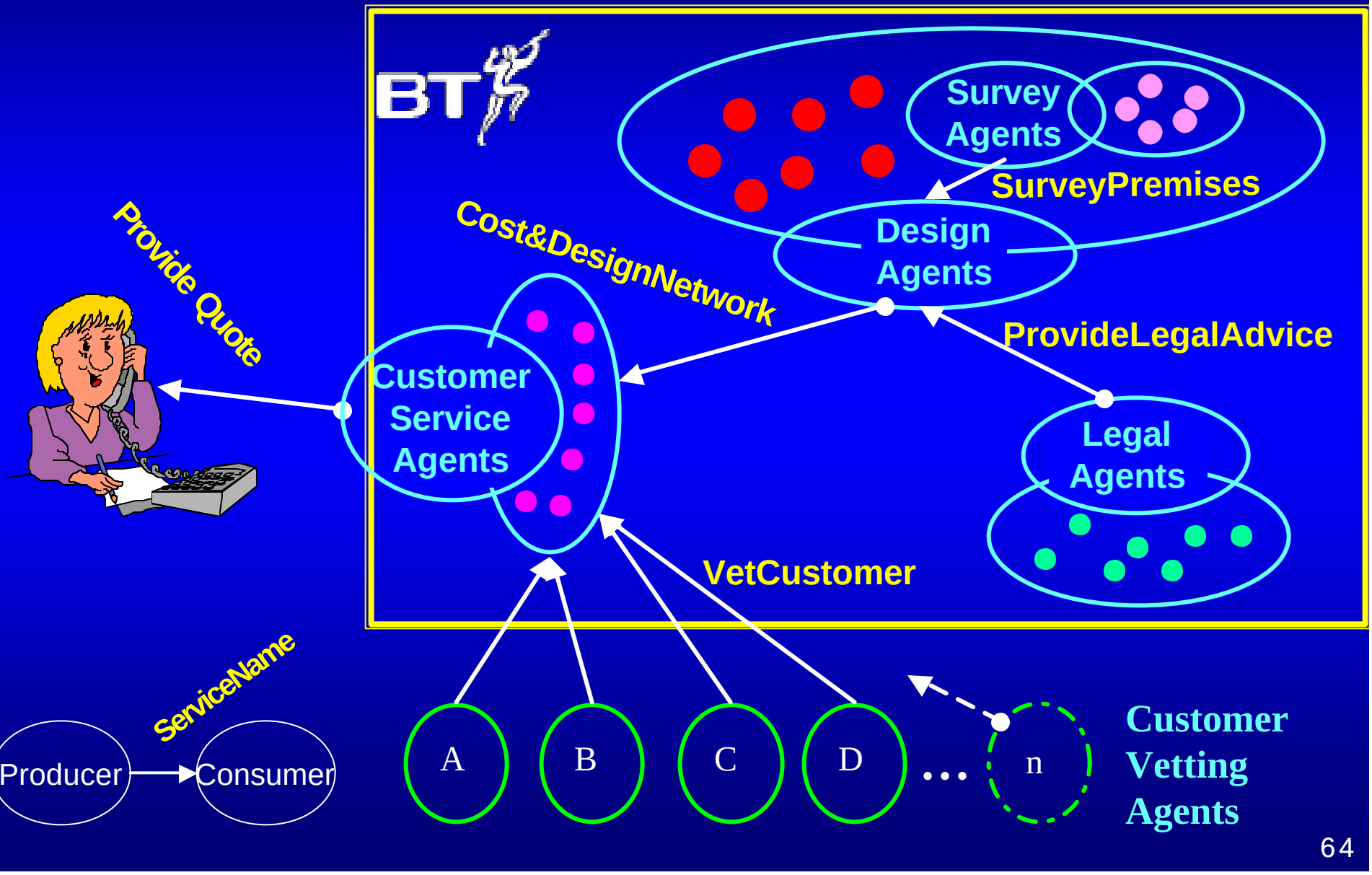
BT's Provide Customer Quote Process

(Jennings *et al.*)



BT's Provide Customer Quote Process

(Jennings *et al.*)



Interaction Engineering:
Service Provisioning = Negotiation

- **Two agents must agree about conditions under which service will be executed**
  - price
  - quality
  - start and end times
  - ……
- **Need to design appropriate negotiation protocol**

The Vet Customer Negotiation

- **Is a 1:many negotiation**
  - One buyer, many sellers
- **Structured as a reverse auction**
  - Efficient means for allowing agent to quickly find partner with highest valuations
  - Multi-dimensional English (open cry) auction

(Vulkan and Jennings, 00)

Multi-Dimensional English Auction

Initiation

Buyer announces

- declared utility function (U)
- maximum price will pay (P)
- minimum acceptable price
- time will wait between offers (T)
- minimum percentage increase that next offer must exceed (X)

can lie about U and P

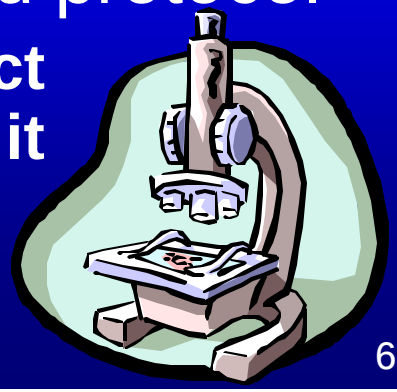
Auction

- **Sellers submit offers (on all dimensions)**
  - can lie and speculate
- **Offer accepted if:**
  - protocol has not terminated
  - offer exceeds last accepted one by X%
- **Buyer makes acceptable bid public**
- **Terminates T seconds after last acceptable offer**

Analysis of the Protocol

Can prove that:

- negotiation will terminate
- no point in buyer lying about U and P
  - **truth telling is dominant strategy**
- no point in seller speculating about competitors
  - **bid X% more than current price, up to reservation level is the dominant strategy**
- in buyer's best interest to use suggested protocol
  - **no other protocol (either auction or direct negotiation) can yield a better result for it**



Organisation Engineering

- **Survey Dept is part of Design Division**
  - Have common goals and objectives
  - Design agents have degree of power over survey agents
    - **Relationship explicitly represented in both agents**
    - **Survey agents still autonomous though!**
- **Impact of organisational relationship**
  - Negotiation is cooperative in nature
    - **Don't reject requests unless cannot meet them**
    - **Quickly move to region of agreement**
      - Agreement produced if one exists
    - **Search for win-win negotiation solutions**
      - Using concept of fuzzy similarity (Faratin et al., 2000)

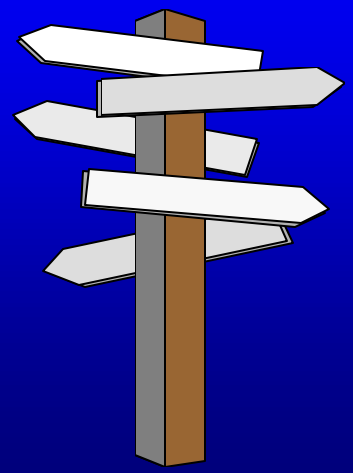
Talk Outline

I. The Essence of Agent-Based Computing

II. Promise of Agent-Oriented Software Engineering

III. Potential Drawbacks

IV. Conclusions



"We seldom attribute good sense, except with those who agree with us."

Duc de la Rouchefoucauld

Promise

Agent-based computing can:

provide powerful metaphors, concepts and techniques for conceptualising, designing and implementing complex distributed systems

Realising the Promise

- **Practical methodologies**
  - For analysing and designing agent-based systems
  - Should be useable by practitioners
- **Industrial strength toolkits**
  - So that don't have to start from scratch
- **Re-useable know-how and technologies**
  - Libraries of interactions protocols
  - Organisational patterns

Acknowledgements

- Cristiano Castelfranchi
- Ed Durfee
- Les Gasser
- Carl Hewitt
- Mike Huhns
- Vic Lesser
- Joerg Müller
- Simon Parsons
- Van Parunak
- Carles Sierra
- Mike Wooldridge
- Franco Zambonelli

References

- P. Faratin, C. Sierra and N. R. Jennings (2000) "Using similarity criteria to make negotiation trade-offs" *Proc. 4th Int. Conf on Multi-Agent Systems*, Boston, 119-126.
- N.R. Jennings (2000) "On Agent-Oriented Software Engineering" *Artificial Intelligence* **117** (2) 277-296.
- N. R. Jennings, P. Faratin, T. J. Norman, P. O'Brien and B. Odgers (2000) "Autonomous agents for business process management" *Int. J. of Applied Artificial Intelligence* **14** (2) 145-190.
- N. Vulkan and N. R. Jennings (2000) "Efficient mechanisms for the supply of services in multi-agent environments" *Int J. of Decision Support Systems* **28** (1-2) 5-19.