QoSPL: A QoS-Driven Software Product Line Engineering Framework for Distributed Real-time and Embedded Systems

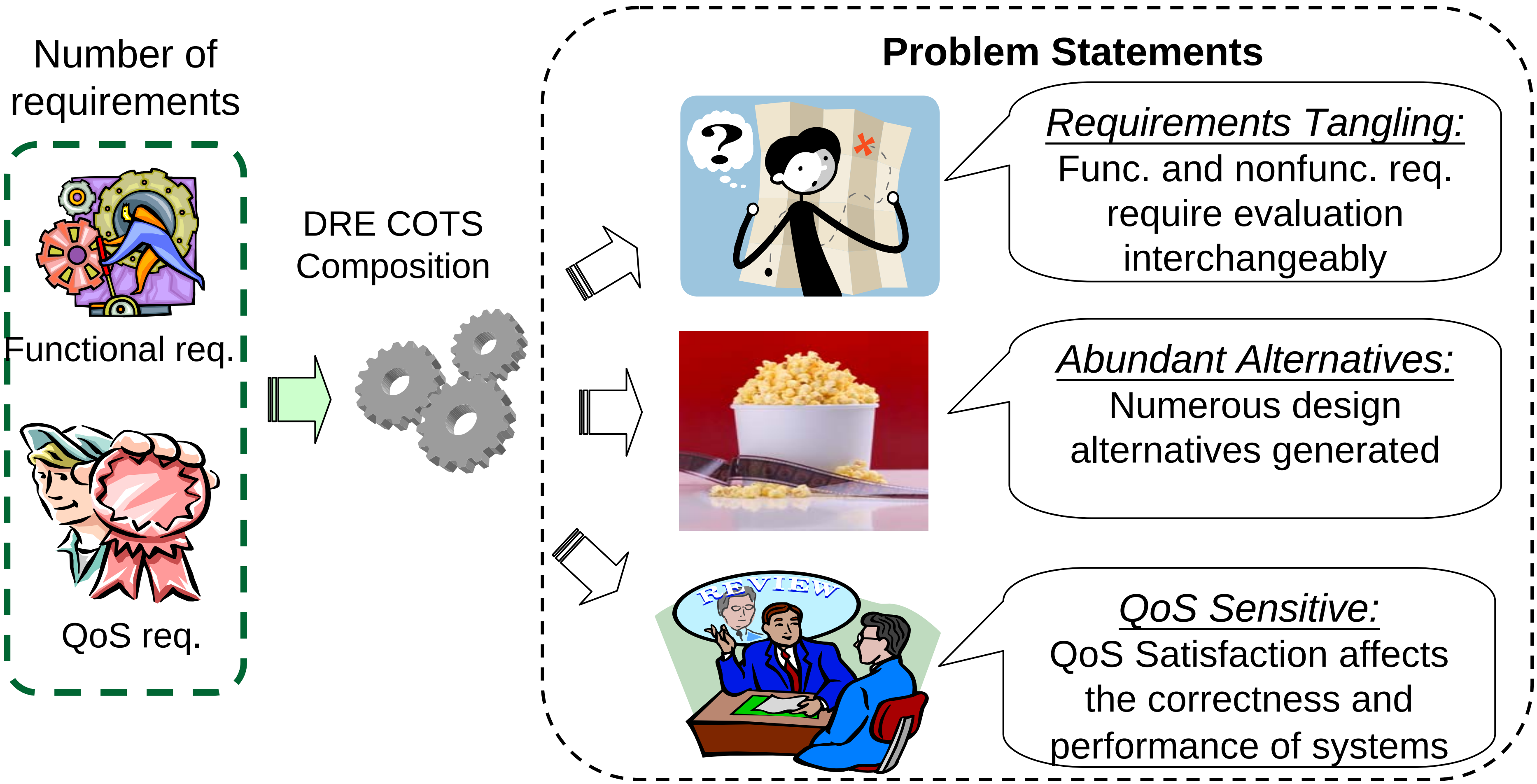
Shih-Hsi "Alex" Liu[1], Barrett R. Bryant[1], Jeff Gray[1], Rajeev Raje[2], Mihran Tuceryan[2], Andrew Olson[2], and Mikhail Auguston[3]

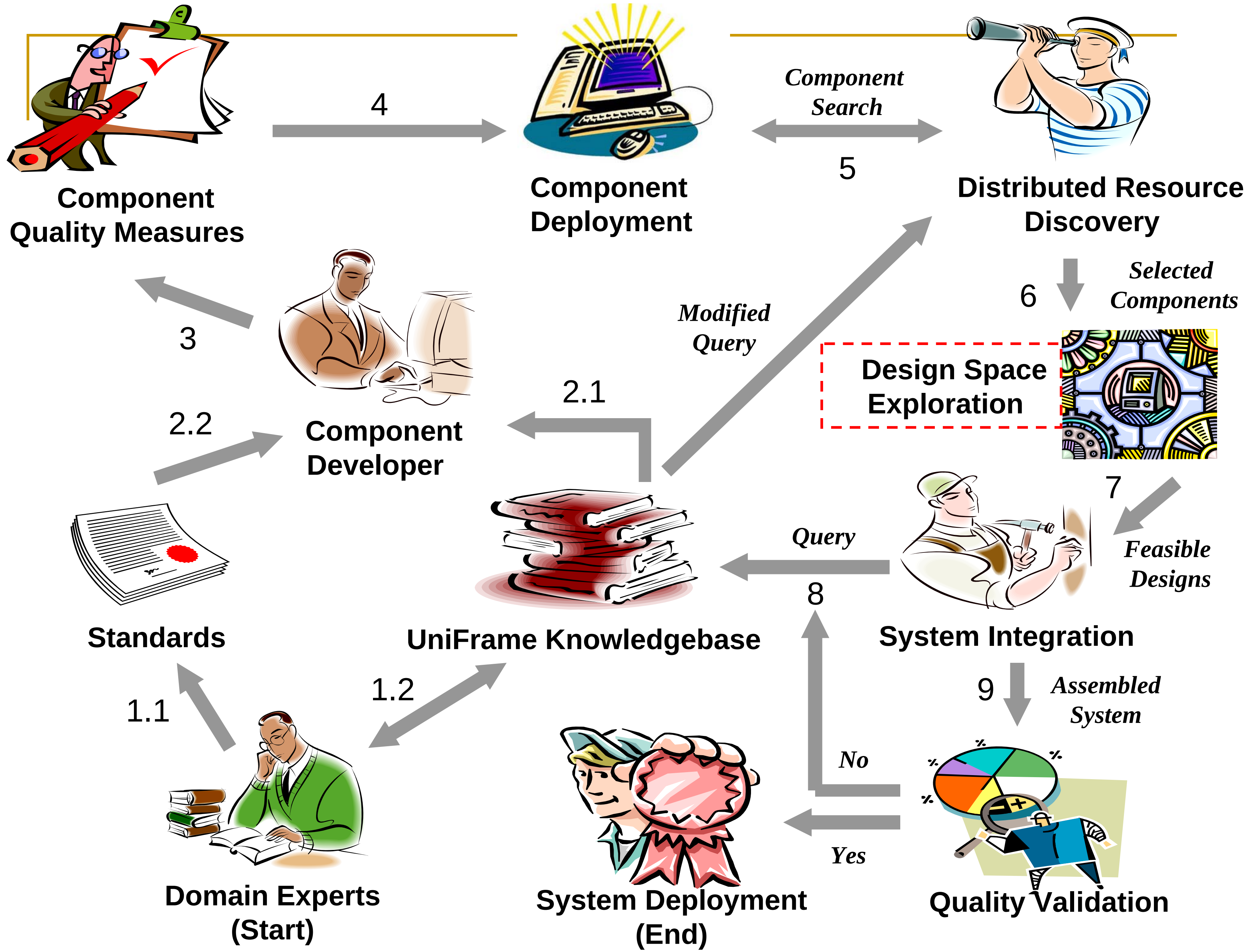
1. University of Alabama at Birmingham
2. Indiana University-Purdue University Indianapolis
3. Naval Postgraduate School

Outline

- Problem Statements
- Background: UniFrame
- QoSPL: a QoS-driven Product Line framework
  - A Case Study: a battlefield training system
  - Domain Engineering (a TLG++ approach)
  - Application Engineering (a Petri Net appoach)
- Related Work
- Conclusion

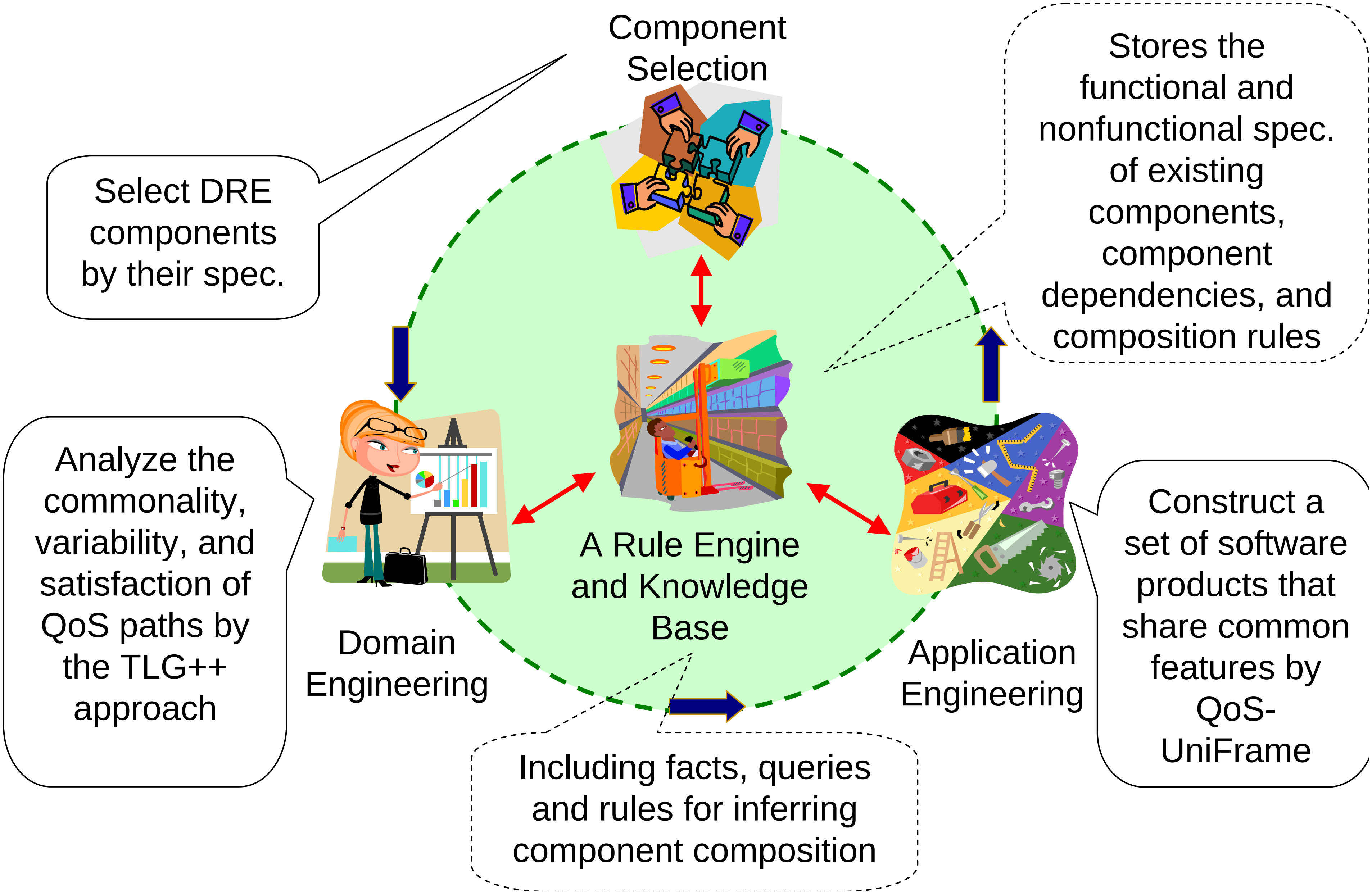
Problem Statements





Component Quality Measures
4
Component Deployment
Component Search
5
Distributed Resource Discovery
6
Selected Components
3
Component Developer
Modified Query
2.1
2.2
Design Space Exploration
7
Feasible Designs
Standards
UniFrame Knowledgebase
Query
8
System Integration
9
Assembled System
1.1
1.2
No
Yes
Domain Experts (Start)
System Deployment (End)
Quality Validation

Overview of the QoSPL

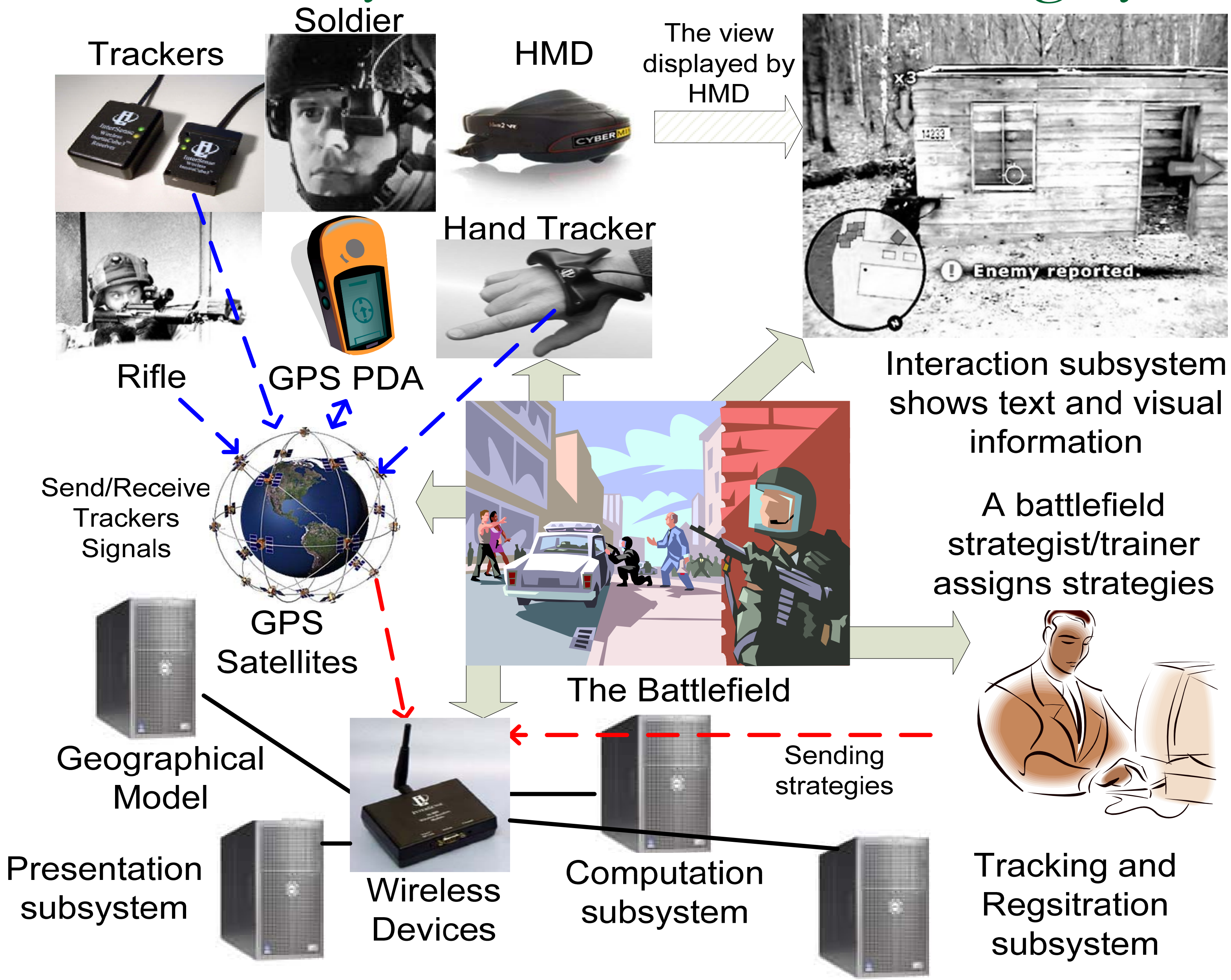


Mobile Augmented Reality Systems

- A DRE system concentrating on enriching the user environment by merging ***real*** and ***virtual*** objects

- Six subsystems:
  - **Computation**: performs specific functionalities for the application
  - **Presentation**: computes virtual multimedia objects
  - **Tracking and registration**: tracks user's position and orientation and registers virtual objects
  - **Environmental model**: store the geometrical and detailed hierarchical 3D information
  - **Interaction**: coordinates virtual multimedia objects
  - **Wireless communication**: provides mobile communications

A Case Study: A battlefield training system



- *A soldier is to rescue a virtual hostage in a battlefield. The position and orientation sensors on his body send back the 6 Degrees Of Freedom (6DOF) data to the tracking subsystem every half second via wireless communication. As the soldier is standing on specific positions with specific orientations in some buildings derived from a predefined tactical scenario stored in the computation subsystem, his HMD displays the enemies registered by the tracking system, rendered by the presentation subsystem and coordinated by the interaction subsystem. The soldier communicates with the command center via his headphone. The information of the soldier's current position is displayed on the HMD by text.*

Domain Engineering: The Two-Level Grammar++ approach

- Goals:
  - Analyze common and variable requirements
  - Design a prescribed reference architecture for its product line
  - Implement a set of reusable core assets
  - Verify and validate the core assets

Domain Engineering: The Two-Level Grammar++ approach

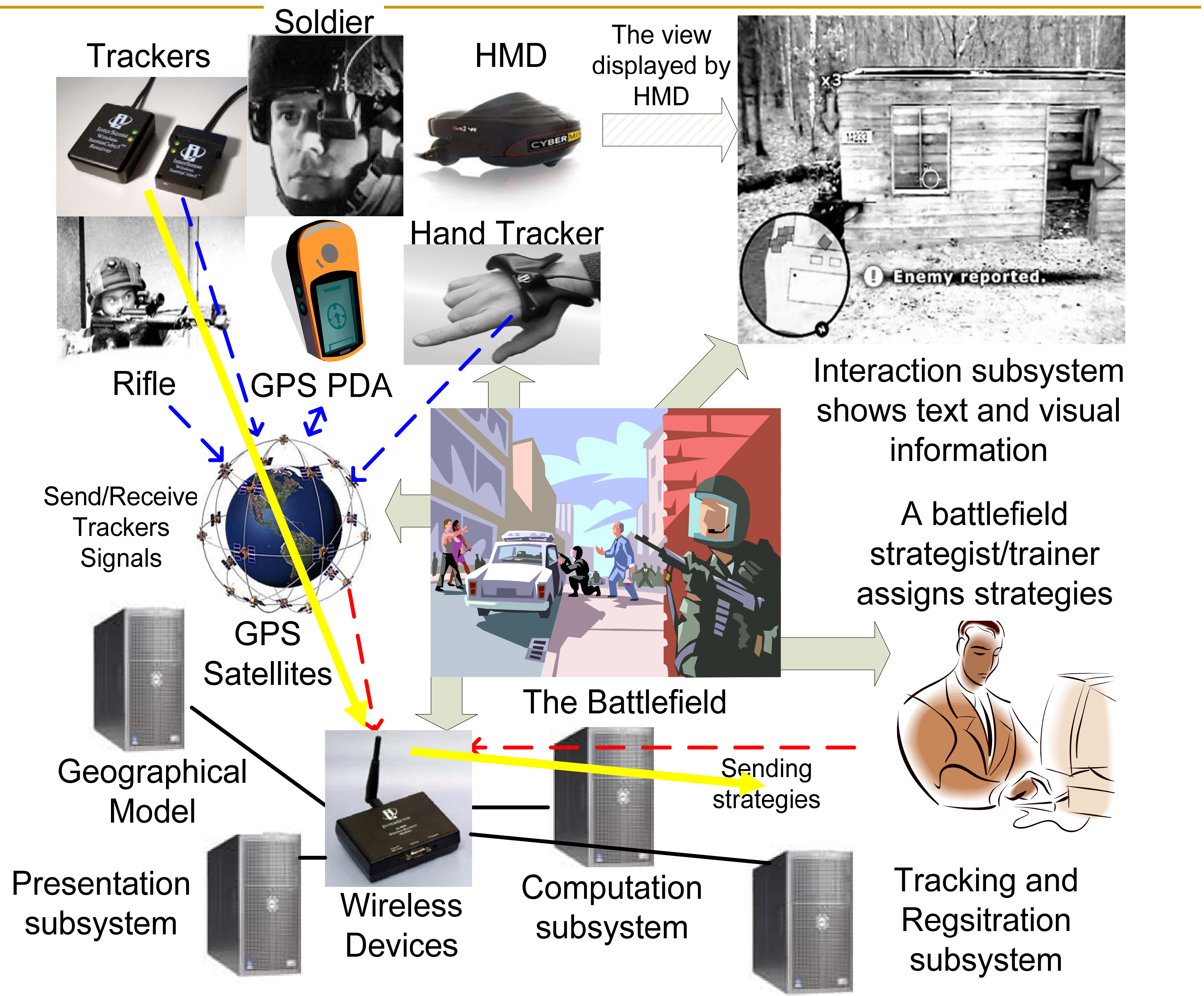
Original TLG++

- Two Context Free Grammars (CFGs):
  - The 1^{st} CFG: Define a set of Parameters
  - The 2^{nd} CFG: Define a set of Function Definitions
- A Grammatical Concept:
  - Define Syntax and Semantics of Programming Languages
    - The 1^{st} CFG: Define the Syntax by Production Rules
    - The 2^{nd} CFG: Define the Semantics of the Production Rules

Domain Engineering: The Two-Level Grammar++ approach

QoS-driven TLG++

- QoS-driven TLG++ specifies and analyzes QoS paths at the service level
- The first CFG utilizes Extended Backus-Naur Form (EBNF) to define the components and direction of a QoS systemic path
  - EBNF represents ***mandatory, alternative, optional,*** and ***OR*** features



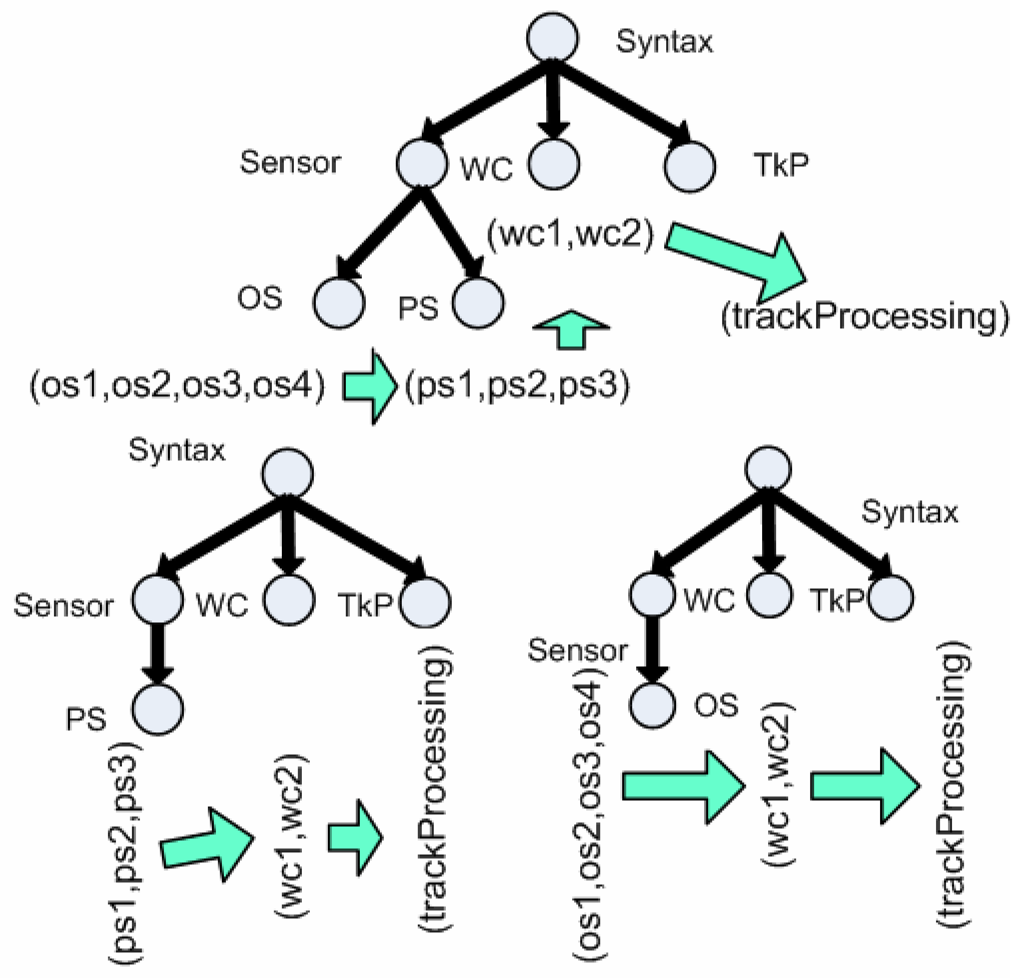
Soldier
Trackers
HMD
The view displayed by HMD
x3
Enemy reported.
Hand Tracker
Rifle
GPS PDA
Interaction subsystem shows text and visual information
Send/Receive Trackers Signals
A battlefield strategist/trainer assigns strategies
GPS Satellites
The Battlefield
Geographical Model
Sending strategies
Presentation subsystem
Wireless Devices
Computation subsystem
Tracking and Regsitration subsystem

TLG++ – The 1st CFG

```
class TextDeadlineFromClient.
   Syntax :: Sensor WC TkP.
   Sensor :: OS PS ; PS ; OS.
   PS :: ps1 ; ps2 ; ps3.
   OS :: os1 ; os2 ; os3 ; os4.
   WC :: wc1 ; wc2.
   TkP :: trackProcessing.
   Sum :: Double.
 semantics of sendPositionFromClient :
      //coming up……..
end class.
```

WC: Wireless communication. TkP: Track and registration process. OS: Orientation sensor. PS: Position sensor.

Domain Engineering



TLG++: The 2nd CFG

- The second CFG defines component dependencies, composition rules, and QoS satisfaction formula
  - Component dependencies: the relationships between components in terms of function-determined and application-specific tasks
  - Composition rules: verify interface consistency between components and pre- and post-conditions of composition by inferences
  - QoS satisfaction formula: quantitatively estimate the satisfaction of the QoS property of a QoS systemic path

TLG++ – The 2nd CFG (cont.)

```
class TextDeadlineFromClient.
 Syntax :: Sensor WC TkP.
 ….
 semantics of sendPositionFromClient :
            PreCondition := semantics of queryComponent with OS PS WC and TkP,
            if PreCondition then
                    Sum := semantics of sumOfMTAT with OS PS WC and TkP,
            else ErrorMessage, end if,
            PostCondition := semantics of queryPattern with Sum.
            Double semantics of sumOfMTAT with OS PS WC and TkP :
                    return OS semantics of getMTAT + …….//[SEKE'05]
end class.
```

Pre-Condition and Post-Condition can be used for validating QoS analysis and eliminating infeasible ones

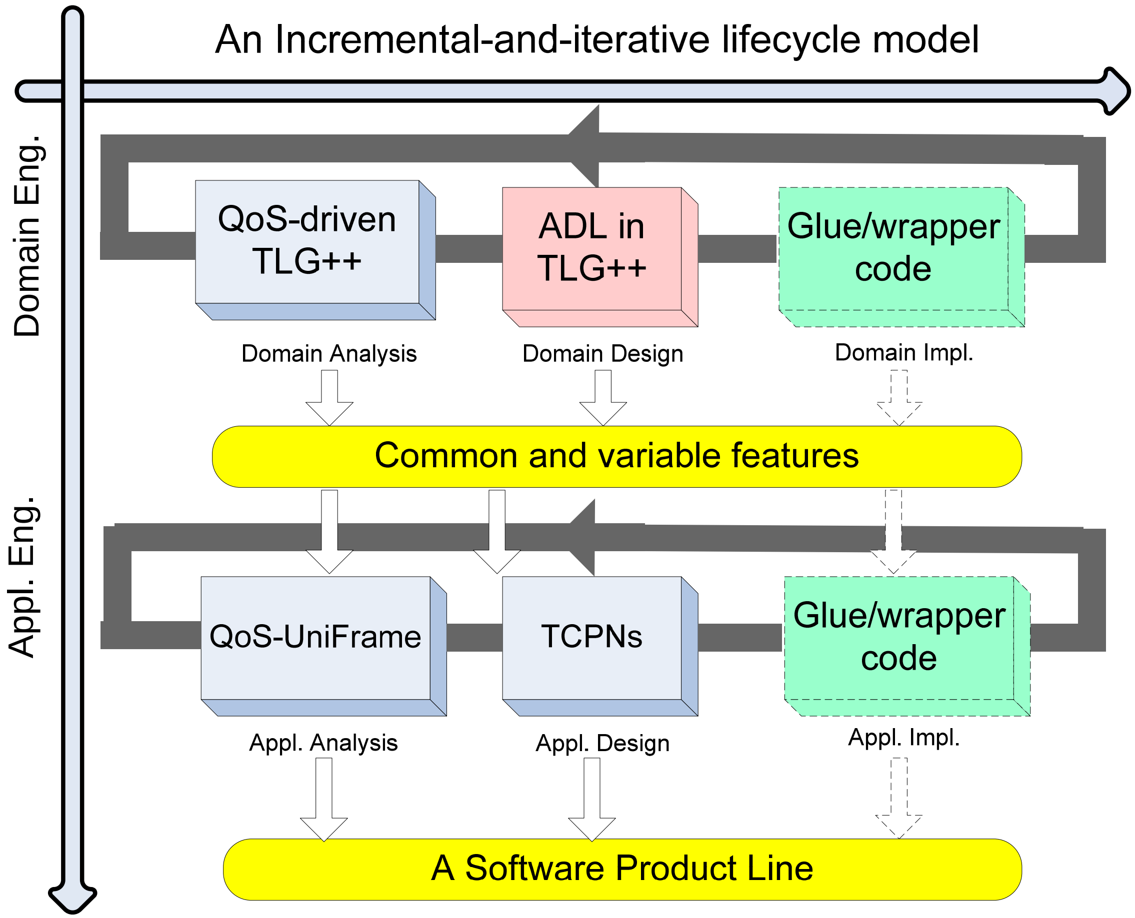
[SEKE'05] S.-H. Liu, et al. Quality of Service-driven requirements analyses for Component Composition: A Two-Level Grammar++ Approach

TLG++: The 1st CFG

- Symbol tables are utilized for analyzing the rate of ***commonality*** and ***reusability*** of QoS systemic path families
- Intuition: More satisfactory QoS paths have higher probabilities to be selected
- [ps2, wc2, tkp]
- [os1, ps2, wc2, tkp] or [os4, ps2, wc2, tkp]

ID	os1	os2	os3	os4	ps1
Type	OS	OS	OS	OS	PS
MTAT (*ms*)	0.2	0.3	0.45	0.2	0.25
ID	ps2	ps3	wc1	wc2	tkp
Type	PS	PS	WC	WC	Tkp
MTAT (*ms*)	0.15	0.2	0.5	0.4	0.9

Overview of the QoSPL



Application Engineering: The Timed Colored Petri Nets Approach

- Goals:
  - Reuse core assets in each workflow in the domain engineering process
  - Exploit variable features of each workflow on individual appliations
  - In our approach: assemble *all* common QoS paths and specific variable paths for different product line members

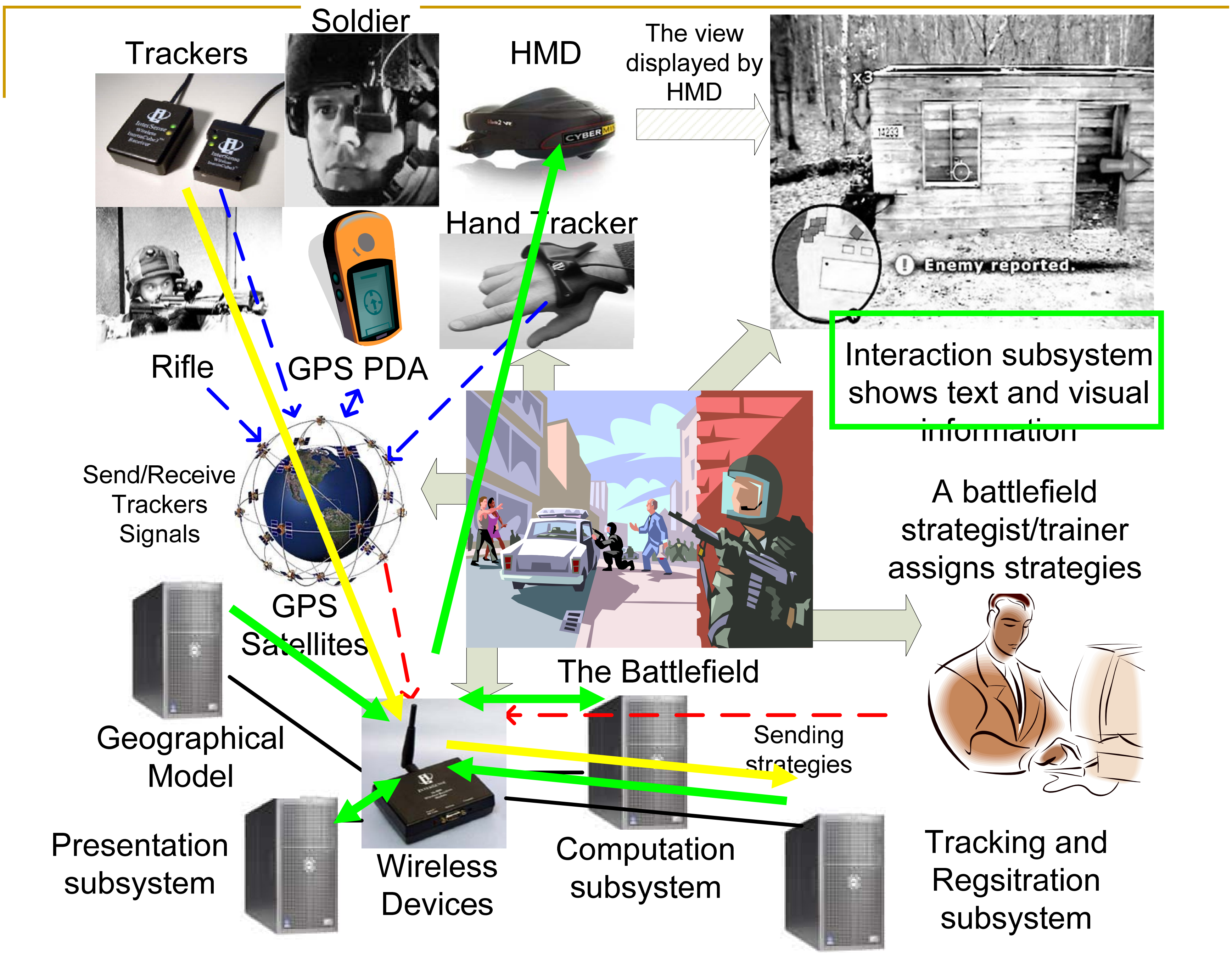
Timed Colored Petri Nets

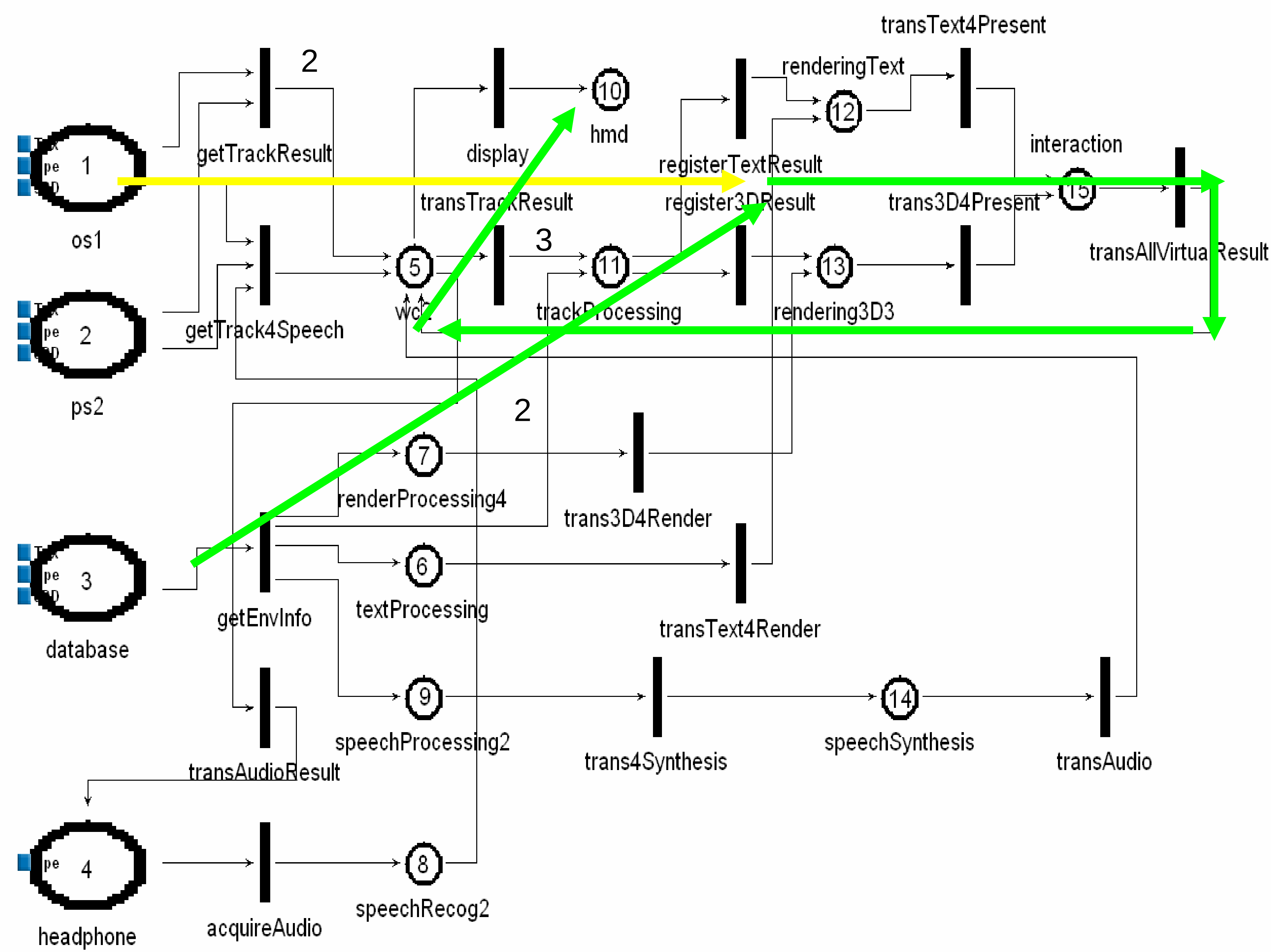
Original Timed Colored Petri Nets (TCPNs)

- Places: states
- Transitions: actions or events
- Arcs: directions (P->T or T->P)
- Tokens: notations specifying configurations of places
- Weights: notations specifying configurations of transitions
- Colors: identities of places
- Time: timer
- Markings: configurations comprising tokens

QoS-driven Timed Colored Petri Nets

- Places: components or sub-components
- Transitions: design decisions (when, what, how) or method calls
- Arcs: directions (P->T or T->P)
- Tokens: QoS parameters or events
- Weights, Colors, Time, Markings: same as above


Soldier
Trackers
HMD
The view displayed by HMD
Enemy reported.
Hand Tracker
Rifle
GPS PDA
Interaction subsystem shows text and visual information
Send/Receive Trackers Signals
GPS Satellites
A battlefield strategist/trainer assigns strategies
The Battlefield
Geographical Model
Sending strategies
Presentation subsystem
Wireless Devices
Computation subsystem
Tracking and Regsitration subsystem

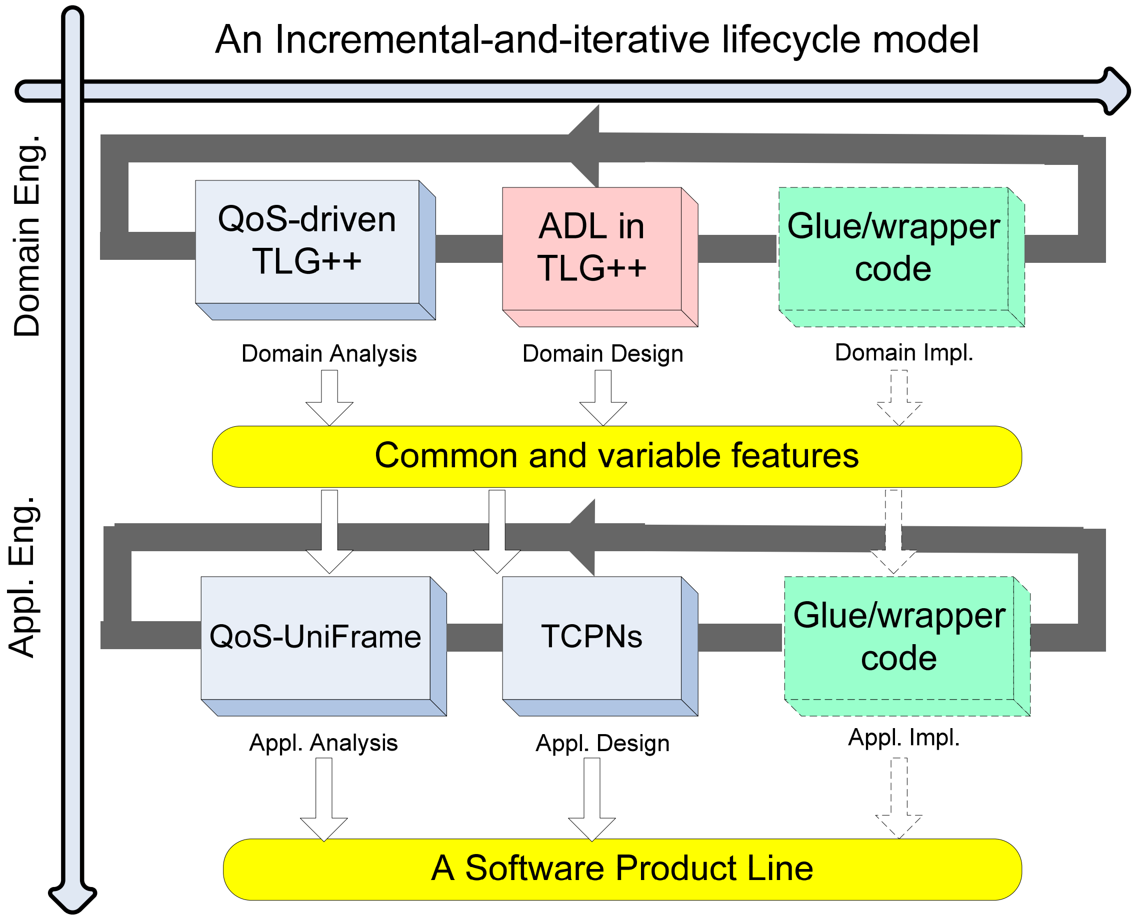


2
3
2
transText4Present
renderingText
hmd
getTrackResult
display
registerTextResult
interaction
register3DResult
trans3D4Present
transTrackResult
transAllVirtualResult
os1
rendering3D3
trackProcessing
getTrack4Speech
ps2
renderProcessing4
trans3D4Render
getEnvInfo
textProcessing
transText4Render
database
speechProcessing2
trans4Synthesis
speechSynthesis
transAudio
transAudioResult
acquireAudio
speechRecog2
headphone

Application Engineering

- The reachability tree analysis
  - Dynamic analysis: evaluate every intermediate node of the tree.
  - Discard the entire branch if an intermediate node is not satisfied
- Marking (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15)
  M0 (3,3,3,1,0,0,0,0,0,0,0,0,0,0,0)
  M1 (1,1,3,0,2,0,0,1,0,0,0,0,0,0,0)
  M2 (0,0,3,0,3,0,0,0,0,0,0,0,0,0,0)
  M3 (0,0,3,0,0,0,0,0,0,0,3,0,0,0,0)
  M4 (0,0,0,0,0,1,1,0,1,0,2,0,0,0,0)
  M5 (0,0,0,0,0,0,0,0,0,0,2,1,1,1,0)
  M6 (0,0,0,0,1,0,0,0,0,0,0,0,0,0,2)
  M7 (0,0,0,0,3,0,0,0,0,0,0,0,0,0,0)
  M8 (0,0,0,1,0,0,0,0,0,2,0,0,0,0,0)

Overview of the QoSPL



Related Work

- FORM: extension of FODA
- KobrA: MDA+ CBSE + SPLE
- QADA: scenario based quality analysis
- Mini-Middleware: customized middleware based on QoS demands

Conclusion

- A separation of concerns approach to focus on QoS properties during the construction of DRE systems
- A DRE product line is assembled by combining different QoS paths
- Solve the three challenges inherent in DRE system construction using traditional CBSE and SPLE
  - QoS sensitive, Requirements tangling, Abundant alternatives
- Two formalisms to facilitate high confidence system construction
- TODO:
  - commonality and variability management utilized by the symbol table
  - Finer-grained QoS analysis
  - Implementation and testing workflows + ADL

Questions

- More information
  - UniFrame: www.cs.iupui.edu/uniFrame
  - QoSPL: www.cis.uab.edu/liush/QoSPL.htm