Next Generation Software Engineering

Ali Mili

RST, Ada Europe 2007,
Geneva Switzerland
June 28, 2007

Background

ULS: Ultra Large Scale Systems

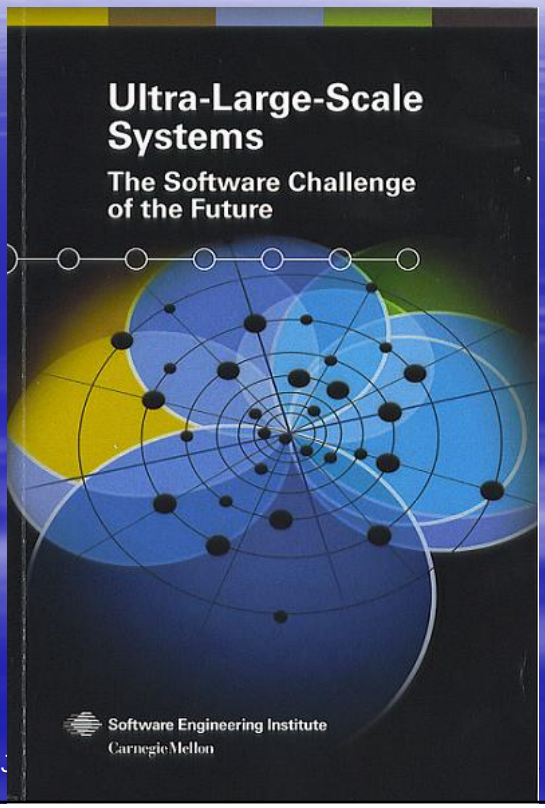
- One year study conducted at SEI: August 2005 to June 2006.
- Goal: Research Roadmap to prepare for the advent of ULS systems.
- SEI Staff: Linda Northrop (leader), Peter Feiler, John Goodenough, Rick Linger, Tom Longstaff, Rick kazman, Mark Klein, Mark Pleszkoch, Kurt Wallnau.

Background: External Panel

- ***Gregory Abowd***, Ga Tech
- ***Carliss Baldwin***, Harvard
- ***Robert Balzer***, Teknowledge
- ***Richard Gabriel***, Sun Microsystems Inc.
- ***Gregory Kiczales***, UBC
- ***John Lehoczky***, CMU
- ***Ali Mili***, NJIT/ SEI
- ***Peter Neumann***, Stanford Research
- ***Mary Shaw***, CMU
- ***Daniel Siewiorek***, CMU
- ***Jack Whalen***, Xerox PARC

Outcome

- *Launched August 2005, meeting in Pittsburgh, PA*
- *ULS Characteristics, Research Challenges, Research Areas.*
- *Three groups, ULS design, ULS quality, ULS operation*
- *Coordination meeting, November 2005.*
- *Write-ups, internal reviews, External reviews.*
- *Final report, June 2006.*
- *Exploration of funding Options, opportunities.*



Outline

- Characteristics of ULS Systems
- Challenges in ULS Systems
- ULS Research Areas

Characteristics of ULS Systems

ULS Systems: One billion lines of code….

Currently, we build/ compose/ maintain/ configure million LOC systems. A difference of scale.

Scale Changes everything!!!

Analogy: a home vs a skyscraper. Not a bigger home, but a totally different object.

ULS Characteristics

- Decentralization (data, development, evolution, operational control).
- Conflicting Requirements (many stakeholders, with distinct, inconsistent needs).
- Continuous evolution and deployment (evolutionary lifecycle).
- Heterogeneous, inconsistent, and changing components (overlap, vs partition).

ULS Characteristics

- Erosion of the people system boundary (people as elements of the system). Also, erosion of developer/ user distinction.
- Normal failure (hardware/ software failures are the norm rather than the exception).
- New paradigms for acquisition and policy (new methods of acquisition/ operation/ control).

Decentralized Control

- Scale precludes centralized/ hierarchical control of data, development, evolution.

Characteristic undermines the following assumption:

- All conflicts must be resolved, uniformly.

Conflicting Requirements

- Scale precludes waterfall-like lifecycle, where requirements are pinned down before development.

Characteristic undermines the following assumptions:

- Requirements can be known in advance and change slowly as system evolves.
- Tradeoff decisions will be stable.

Continuous Evolution and Deployment

- Indefinite evolution, no phaseout.
- Distributed evolution, at the whim of a broad class of users/ stakeholders/ administrators.
- Evolution: distributed, constrained change rather than centrally planned change (free market vs centralized economy).

Characteristic undermines the following assumption:

- System improvements are introduced at discrete intervals (build-use-build).

Heterogeneous, Inconsistent, Changing Components

- Heterogeneous: distinct sources, platforms, languages, methods, etc…
- Inconsistent: developed independently, no mutual cognizance, generating unpredictable / unpredicted emergent behavior.
- Changing: in non centrally planned nor centrally observable manner.

Heterogeneous

Characteristic undermines the following assumption:

- Effects of change can be predicted or modeled.
- Configuration information tightly controlled.
- Components and users homogeneous.

Systems are not partitioned into components; rather components overlap. Alters composition rules profoundly.

Erosion of the People System Boundary

- People as part of the system (decision nodes). All system modeling profoundly altered.
- The law of large numbers allows for analysis of system observation that is alien to small systems (Thermodynamics vs Particle Physics).

Characteristic undermines the following assumptions:

- People are just users of the system.
- Collective people behavior of no interest.
- Social interactions irrelevant.

Normal failure

- Excessive Size, Complexity, Heterogeneity ➔ Pervasive failures throughout
- Change of focus: no longer on avoidance, removal, but mostly on recovery/ redundancy.

Characteristic undermines the following assumptions:

- Infrequent failures.
- Defects can be removed.
- Effects can be modeled away.

New Paradigms for Acquisition and Policy

- Distributed acquisition, management, control, administration, evolution.
- Multiple providers, multiple stakeholders, multiple user classes.
- Emphasis on constraining (through rules) rather than proactive decision making.

Characteristic undermines the following assumption:

- A prime contractor is responsible for system development, evolution, operation (impact: liability, organization, procurement).

Challenges in ULS Systems

ULS Characteristics put most current SE knowledge out of commission.

Three broad areas of consideration:

- Design and Evolution (Darwin Team).
- Orchestration and Control.
- Monitoring and Assessment (Intelligent Design Team).

Design and Evolution

- Challenge: harness and coordinate design capabilities and motivation of individual companies, prime contractors, supply chains, but also whole industries, within which competition will drive exploration of richer design spaces.
- Developing and evolving architectures around which industries will organize (re: auto industry).
- Design of the industrial ecosystem, including incentive structures and sources of value.

Design and Evolution

ULS Systems will include not only IT components, but also:

- Machines,
- Individuals and teams,
- Diverse sensors,
- Information streams and stores.

Traditional view: SW as programming the computer. New view: SW as programming all the information processing mechanisms of a ULS system.

Design and Evolution

- ULS systems will require an internal infrastructure that supports development and evolution, initiated by system designers, implementers, operators, users.

Research Challenges in Design and Evolution

- Economics and Industry structures: aligning design architecture and industry structures to harness economic forces to meet key ULS requirements.
- Social activity for constructing computational environments: modeling social interactions to support ULS design.
- Legal Issues: Licensing, intellectual property, liability, certification (safety, security) in a ULS context (self-configurability?).

Research Challenges in Design and Evolution

- Enforcement mechanisms and processes for the set of (legal, design and process) rules that support and maintain the integrity of the system.
- Definition of common services supporting ULS systems: defining an infrastructure (technological, legal, social services) that is common to many elements of ULS systems.
- Rules and Regulations: How will whole industries agree on standards to ensure coherence and quality.

Research Challenges in Design and Evolution

- Agility: ability to respond to changes in process or product parameters.
- Integration: ability to integrate heterogeneous components efficiently and reliably.
- User controlled evolution: Providing components and composition rules that afford users ability to create new capabilities.

Research Challenges in Design and Evolution

- Computer supported evolution: Automated tools to support the development and evolution of ULS systems.
- Adaptable structure: ability to create designs that are effective even as requirements and environments change.
- Emergent quality: ability to organize processes for producing ULS systems so that they converge on high quality designs.

Research Challenge: Orchestration and Control

Orchestration:

- set of activities needed to maintain harmony in the operation of a ULS system.
- Orchestration requires upfront design, overall policy (formulation, enforcement), and real-time parameter tuning.
- Dealing with interdependencies between local, intermediate and global events/ properties (analogy with city).

Research Challenge: Orchestration and Control

We need new knowledge, technologies, methods in:

- Online modification. Parts of the system may be shut down, under repair, under modification… how to ensure this does not affect other parts in undesirable or unpredictable ways.
- Maintaining quality of service. Maintaining overall QoS while meeting local QoS requirements.
- Creation and Execution of Policies and Rules. What policies and rules lead to effective solutions for divergent stakeholders (no zero sum game)? How are such rules promulgated? Enforced?

Research Challenge: Orchestration and Control

- Adaptation to Users and Contexts. Eliciting/ representing/ discovering/ analyzing user needs; catering to these needs at run-time.
- Enabling user controlled orchestration. Providing components and composition rules that give users the ability to adapt and customize portions of the system in the field (blurring the line sys admin/ developer/ maintainer/ end user).

ULS Challenge: Monitoring and Assessment

- Assessing the effectiveness of a ULS system design, orchestration, and evolution.
- Monitor and analyze system state, behavior, health.
- Criteria of good health/ fitness/ adequacy depend on stakeholder (administrator, user, partner, agent), stage of evolution, system component, etc.
- Moving from narrow/ dry metrics based assessment to more global measurement of fitness/ health.

Research Challenge: Monitoring and Assessment

- Scale, decentralization, distribution, heterogeneity of ULS systems challenges current assessment techniques.
- Many measures may be statistical rather than punctual (e.g. GDP).
- New technologies are needed to: define quality expectations; learn ways to maintain a ULS within these expectations; learn to influence evolution of ULS system towards a particular quality goal.

Research Challenge: Monitoring and Assessment

Sample Challenges:

- Defining the indicators: local metrics, global metrics, intermediate metrics (subsystem specific, stakeholder specific).
- Understanding how indicators change: What adjustments have an impact on indicators.
- Prioritizing the indicators. Generality, relative importance, impact, etc.

Research Challenge: Monitoring and Assessment

- Handling change. Measures must be updated as the system evolves, user needs evolve.
- Imperfect information. Information may be inaccurate, stale, partial, etc.
- Gauging the Human element. Indicators of health and performance of people, businesses, organizational components.

Research Areas

Interdisciplinary portfolio of seven research areas, to address the three challenges of ULS system design evolution, orchestration and control, monitoring and assessment.

- ULS characteristics fundamentally undermine the assumptions of today's approaches, breakthroughs are needed.
- Ecosystem nature of ULS systems: a more expansive view of software research, interactions with physical and social sciences.

Seven Research Areas

- ***Human Interaction***. Humans as elements of a socially constituted computational process. Research involves anthropology, sociology, social sciences. Constructing and Evolving socio technical systems.
- ***Computational emergence***. Some aspects of ULS systems are driven by incentivizing and constraining behavior, rather than prescribing behavior. Game theory, mechanism design. Ensuring globally optimized behavior from locally driven (self interest) behaviors.

Seven Research Areas

- ***Design.*** From a technology-centric discipline to a discipline focused on people and organizations. Social, cognitive and economic considerations. Design structures, such as design rules and government policies. Variety of levels of abstraction.

Seven Research Areas

- ***Computational Engineering.*** Turning the development, analysis and evolution of ULS systems and components into a mature engineering discipline. Compiled knowledge captured into tools, methods, techniques.
- ***Adaptive System Infrastructure***. Development environments and run time platforms that support decentralization of ULS systems. Means to enable development of ULS systems in their deployment environment (another blurred distinction: development environment vs operational environment).

Seven Research Areas

- ***Adaptable / Predictable System Quality***. Managing traditional qualities such as security, performance, reliability, usability is necessary but insufficient. Maintaining quality in the face continuous change, ongoing failures and sustained attacks. Identifying, predicting, controlling new indicators of system health.

Seven Research Areas

- ***Policy, Acquisition and Management.*** Replacing traditional procurement processes with acquisition policies that accommodate the rapid/ continuous evolution of ULS systems by treating suppliers and supply chains as components of the ULS system. Another distinction blurred.

Research Areas vs Challenges

Research Area	Design and Evolution	Orchestration Control	Monitoring Assessment
Human Interaction			
Computational Emergence			
Design			
Computational Engineering			
Adaptive System Infrastructure			
Adaptable System Quality			
Policy, Acquisition, Management			

38

Three Areas of Personal Interest

- Quantifying Dependability (ORNL, ULS Quality).
- Modeling System Redundancy (Normal Failure).
- Automated Analysis of Software Product (Computational Support, Next Generation SE).

Quantifying Dependability

A Challenge to MTTF:

- No cognizance of variance with respect to subspecification.
- No cognizance of variance with respect to stakeholder.
- No cognizance of variance with respect to verification and validation.
- No cognizance of variance with respect to operational profiles.

Quantifying Dependability

Mean Failure Cost:

- Quantified by $ per unit of operation time.
- Computed separately for each stakeholder.
- Takes into account variance in stakes associated with subspecifications.
- Takes into account variance in probability of failure with respect to each subspecification.

Quantifying Dependability

$$MFC(St) \leq \sum_{R_i} FC(St, R_i) \times \pi(\overline{S \supseteq R_i}).$$

Stakeholders	*Ord*	*Prm*	*Sort*	Mean Failure Cost	MTTF
Binary Searcher	10	2		0.1002	
Median Finder	4	4		0.0404	
Linear Searcher	7	10		0.0710	
Brute Force Searcher	0	10		0.0010	
Probability of Success Probability of Failure	0.99 0.01	0.9999 0.0001	0.98991 0.01009		MTTF= $T \times 98.6072$

No distinction between reliability and safety. Both captured in MFC.

Quantifying Dependability

Stakes, Stakeholders in a FCS

- Stakeholders: Passenger, Pilot, FAA, Airline Company, Airline manufacturer, Insurance Company…
- Subspecs: Smooth ride, Fuel efficiency, Timeliness, Flight Vector, Safety Reqs,…
- Stakes: loss of life, loss of corporate loyalty, airline reputation, aircraft reputation, insurance claim…
- Applications: Premium per hour vs MFC(Insurer). Income per hour vs MFC(Airline). Design, Verification cost vs MFC(Manufacturer). Business Gain or Leisure vs MFC(Passenger).

Modeling System Redundancy

- Redundancy as a system attribute, alongside modularity, size, complexity. Qualitative and quantitative models.
- More to redundancy than duplicating bits, duplicating components, duplicating variables.
- Despite pervasiveness, poorly defined/ modeled.

External Forms of Redundancy

- ***State redundancy***. More representations than representables.
- ***Functional redundancy***. More functionality than required.
- ***Control redundancy***. More than one way to achieve given effect.
- ***Temporal redundancy***. Relations between successive values.

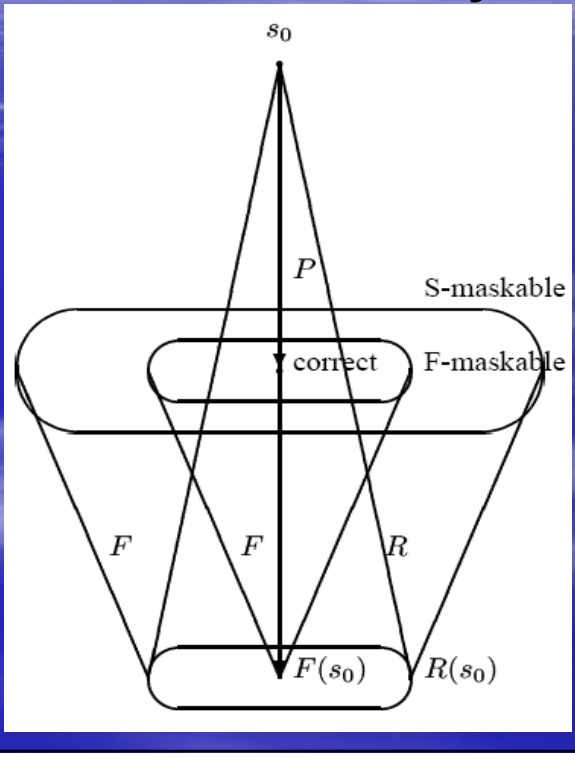
Internal Manifestations of Redundancy

- Non surjectivity of Representation Mappings.
- Non surjectivity of System Functions.
- Non injectivity of System Functions.
- Non determinacy of System Specifications.

Vision: Define redundancy qualitatively, by a general formula, that captures all these attributes.

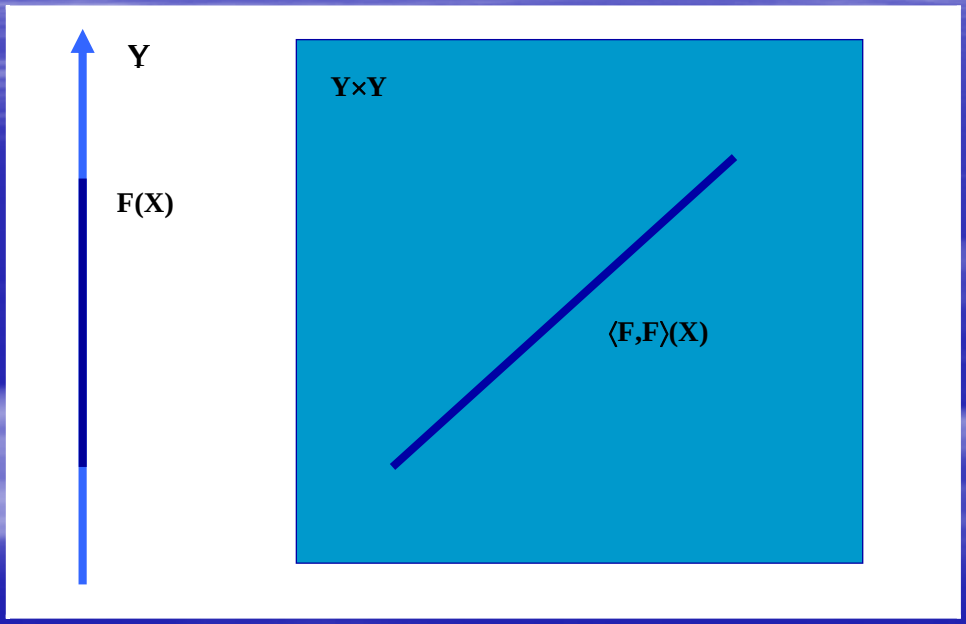
$$rng(R \circ F).$$

Internal Manifestations of Redundancy



47

Redundancy and Duplication



Quantitative Models

- State Redundancy:

$$D(S) = \frac{W(S)}{H(S)} - 1.$$

- Functional Redundancy:

$$D(F) = \frac{H(Y)}{H(F(X))} - 1.$$

State Redundancy

- Positive for any lossless coding.
- Minimal for Huffman coding.
- Increased with non uniform probability distributions.
- Negative for non prefix codes.

Functional Redundancy

Name	Expression	Input Space	Output Space	Redundancy
F_1	X	$B5$	$B5$	0
F_2	$2 \times X$	$B5$	$B5$	0.25
F_3	X **mod** 4	$B5$	$B5$	1.5
F_4	X **mod** 16	$B5$	$B5$	0.25
F_5	X *div* 2	$B5$	$B5$	0.25
G_1	$\langle F_1, F_1 \rangle$	$B5$	$B5^2$	1.0
G_2	$\langle F_2, F_2 \rangle$	$B5$	$B5^2$	1.5
G_3	$\langle F_3, F_3 \rangle$	$B5$	$B5^2$	4.0
G_4	$\langle F_4, F_4 \rangle$	$B5$	$B5^2$	1.5
G_5	$\langle F_1, F_2 \rangle$	$B5$	$B5^2$	1.5

Functional Redundancy

$D(\langle F,F \rangle)=1+2\times D(F).$

Functional Extraction

Deriving the function of a while loop, written in some programming language, say C++. Under the following conditions:

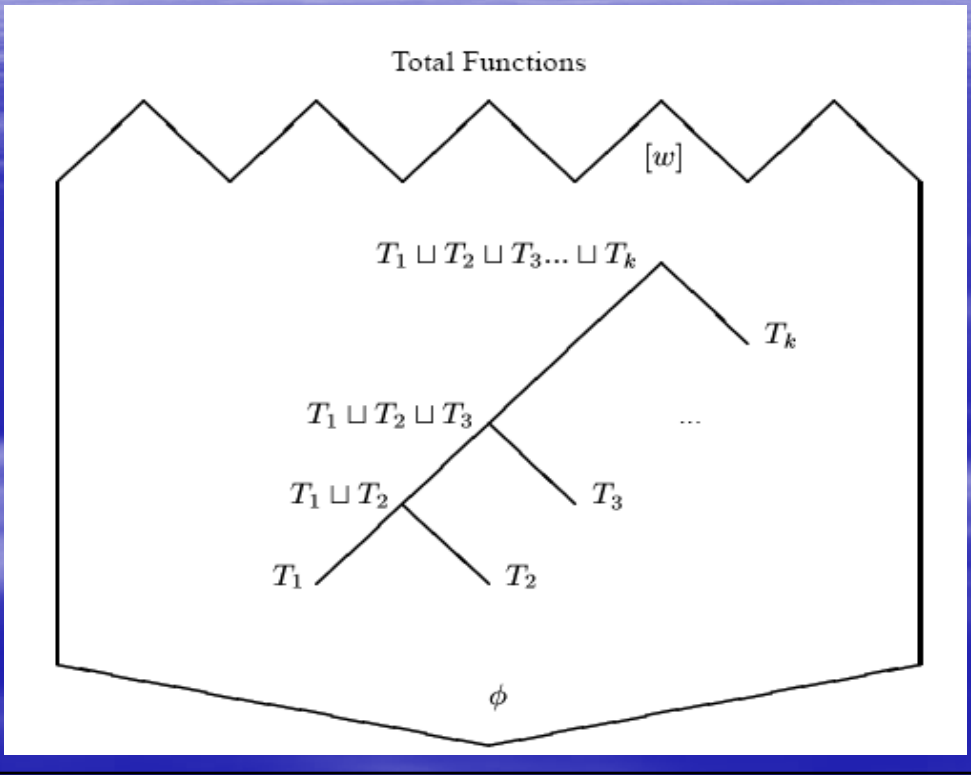
- Closed Form Expression of the loop function (bridging the inductive argument).
- Deriving the Loop Function by Successive Approximations.
- Providing Substitutes for the Loop function. As the extraction capability evolves, we cover more loops, and more of each loop.

Refinement Structure

Refinement Ordering, Refinement Lattice.

- Partial Ordering.
- Any two specs have a meet.
- Join exists conditionally.
- Universal lower bound.
- No universal upper bound.
- Maximal elements: total deterministic.

Structure of the Refinement Lattice



Theorem of Loop Extraction

Given a while loop (*while t do B*), if we find a reflexive transitive relation *R* that is a superset of *B*, then *W* refines R/t.

R/t: lower bound for *W*.

Synopsis of Proposed Approach

Deriving the Loop Function by successive approximations, using lower bounds.

1. How do we find lower bounds?
2. How do we combine them?
3. How do we know we are done?

Function Extraction Algorithm

Three Steps:

1. loop.cpp → loop.cca.
   (conditional) concurrent assignments.
2. loop.cca → loop.mat.
   Inspecting any set of cca, deriving lower bounds.
3. loop.mat → loop.bn
   resolve the equations defined by the lower bounds to derive primed variables.

CCA Form

loop.cpp	loop.cca

while t
{x = x+5;
y = y+x-2;
z = z+y;
x = x+1}
	while t
	{x = x+6,
	y = y+x+3,
	z = z+y+x+3}

CCA Form

If B = {CCA1, CCA2, CCA3, …} then we interpret this in terms of relations as:

$$B = CCA_1 \cap CCA_2 \cap CCA_3 \cap ....$$

Each CCAi represents a superset of B! Each pair of CCAi's represents a superset of B; each triplet of CCAi's represents a superset of B.

Basis of separation of concerns.

Sample 1 Recognizer

- State space: x: int; const c: int;
- Code pattern: x:= x+c.
- Lower bound of [w]:

Sample 1-Recognizer

- State space: x: int; const c: int;
- Code pattern: x:= x+c.
- Lower bound of [w]:

$$R = \{(s, s') \mid x.\mathrm{mod}.c = x'.\mathrm{mod}.c\}.$$

$$V = R \,\mathrm{o}\, I(\neg t).$$

Sample 2-Recognizer

- State Space: x, y: int; const a, b: int.
- Code pattern: x:= x+a, y:= y+b.
- Lower bound of [w]:

Sample 2-Recognizer

- State Space: x, y: int; const a, b: int.
- Code pattern: x:= x+a; y:= y+b.
- Lower bound of [w]:

$$R = \{(s, s') \mid bx - ay = bx' - ay'\}.$$

$$V = R \circ I(\neg t).$$

Sample 2-Recognizer

- State Space: x,y: listtype;
- Code pattern:

  y:= y.head(x);

  x:= tail(x);
- Lower bound of [w]:

Sample 2-Recognizer

- State Space: x,y: listtype;
- Code pattern:

  y:= y.head(x);

  x:= tail(x);
- Lower Bound of [w]:

$$R = \{(s, s') \mid x.y = x'.y'\}.$$

$$V = R \circ I(\neg t).$$

Sample 2-Recognizer

- State space: i: int; x: sometype;
- Code pattern:

  i:= i-1;

  x:= f(x);
- Lower bound of [w]:

Sample 2-Recognizer

- State space: i: int; x: sometype;
- Code pattern:

  i:= i-1;

  x:= f(x);
- Lower bound of [w]:

$$R = \{(s, s') \mid f^{i}(x) = f^{i'}(x')\}.$$

$$V = R \circ I(\neg t).$$

Application

Loop.cpp	Loop.cca

while x!=0
{x = x-1;
y = y+2;
z = z+x+1;
w = w+2*(y-2);}
	while x!=0
	{x = x-1,
	y = y+2,
	z = z+x,
	w = w + 2*y}

Applications III

Loop.mat	Loop.nb
Simplify[reduce[{2*x+y=2*xP+yP, z+(x*(x+1)/2)=zP+(xP*(xP+1)/2), w-(y*(y-2)/2)= wP-(yP*(yP-2)/2), xP=0}, {xP,yP,zP,wP}]]	xP=0 yP=2*x+y wP=w+2*x*(x+y-1) zP=(x+x*x+2*z)/2

Applications, IV

```
{int x;
int t;
int j;
int v;
int w;
const int a;
const int b;
const int c;
const function f;
int y;
int z;

while (x >= 5)
   {y = y+b,
    x = x-5,
    z = z+10,
    j = j-1,
    v = f(v),
    w = w + v,
    t = t-c}
}
```

```
Simplify[ Reduce[ {

%%----------One-Recognizers--
Mod[y,|b|] = Mod[yP,|b|],
Mod[x,|5|] = Mod[xP,|5|],
Mod[z,|10|] = Mod[zP,|10|],
Mod[j,|1|] = Mod[jP,|1|],
Mod[t,|c|] = Mod[tP,|c|],

%%----------Two-Recognizers-
b*x+5*y = b*xP+5*yP,
b*z-10*y = b*zP-10*yP,
10*y-b*z = 10*yP-b*zP,
b*j+1*y = b*jP+1*yP,
b*t+c*y = b*tP+c*yP,
10*x+5*z = 10*xP+5*zP,
5*j-1*x = 5*jP-1*xP,
1*x-5*j = 1*xP-5*jP,
5*t-c*x = 5*tP-c*xP,
c*x-5*t = c*xP-5*tP,
10*j+1*z = 10*jP+1*zP,
10*t+c*z = 10*tP+c*zP,
1*t-c*j = 1*tP-c*jP,
c*j-1*t = c*jP-1*tP,
```

```
%%----------Three-Recognizers---

Nest[f,v,j] = Nest[f,vP,jP],
w+Sum[Nest[f,v,k],{k,1,j}] =
wP+Sum[Nest[f,vP,k],{k,1,jP}],

%%----------Last State Satisfies Not t--

Not(xP>=5),

%%----------Penultimate State Satisfies t-

Resolve [ Exists [
{jPP,tPP,vPP,wPP,xPP,yPP,zPP},
(xPP>=5) &&
tP=tPP-c &&
wP=wPP+vPP &&
vP=f(vPP) &&
jP=jPP-1 &&
zP=zPP+10 &&
xP=xPP-5 &&
yP=yPP+b] ],

{jP, tP, vP, wP, xP, yP, zP} ]]
```

Conclusion

ULS Systems

- Ultra Large, but also…
- Distributed, Decentralized, Evolving, Heterogeneous, Failure prone, Human dependent.
- Intermediate between centralized small systems and loosely coupled systems.
- Challenges current technical capabilities, calls for radically new organizational paradigms.

Further information: http://www.sei.cmu.edu/uls/