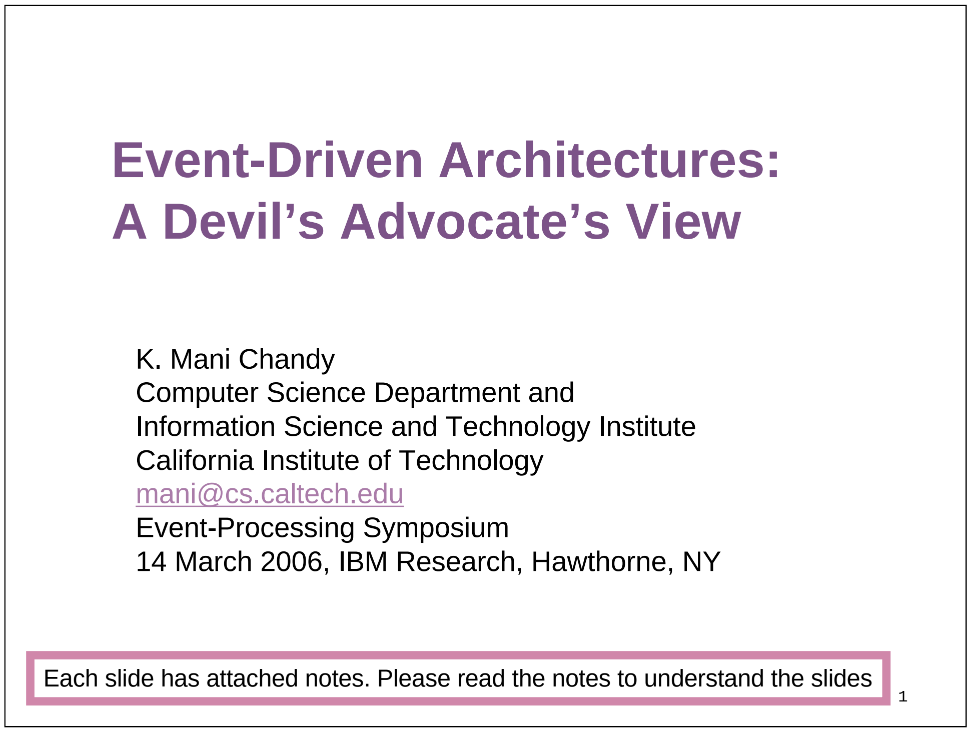


Event-Driven Architectures: A Devil's Advocate's View

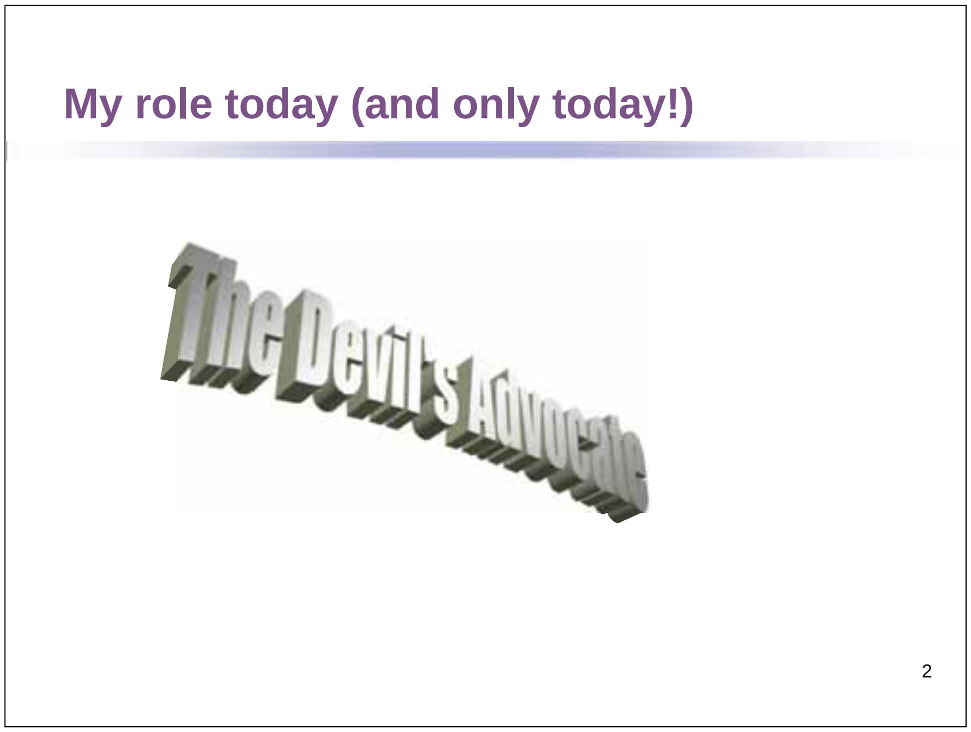
K. Mani Chandy
Computer Science Department and
Information Science and Technology Institute
California Institute of Technology
mani@cs.caltech.edu

Event-Processing Symposium
14 March 2006, IBM Research, Hawthorne, NY

Each slide has attached notes. Please read the notes to understand the slides

1

Notes attached to most slides.

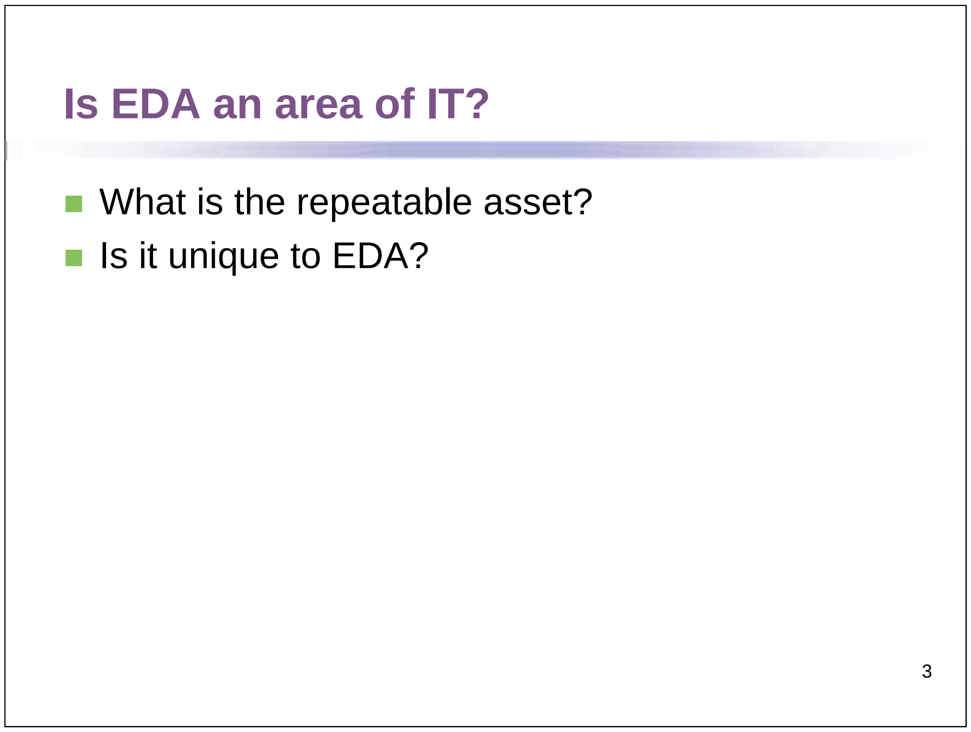


My role today (and only today!)

2

Before a person achieves sainthood, a devil’s advocate presses the case against the candidate. I want to canonize “Event-Driven Architectures” just as you do. But before we do that we need a devil’s advocate to press the case against EDA. Perhaps this negative case will help engender a discussion in which the value offered by different aspects of EDA become clear.

Remember, I’m on your side. I’m playing devil’s advocate just for the day.



Is EDA an area of IT?

- What is the repeatable asset?
- Is it unique to EDA?

3

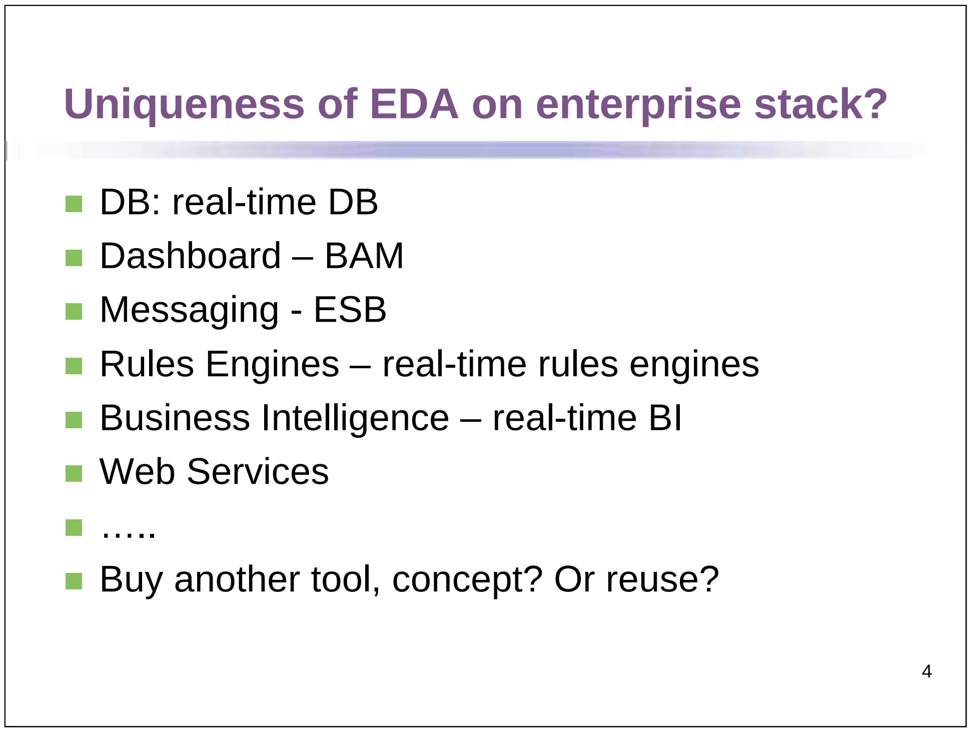
An area of technology is defined by an asset that can be used repeatedly to solve problems in some large space. What is the repeatable asset for EDA? If there are such assets, are they better in EDA than in other products in the enterprise stack?

Assets in a technology often start with ideas for a common abstraction, methodologies for problem solving, best practices in professional services, and tools. Which of these do we have for EDA?

A fundamental difficulty with our area is that we aren't clear about what problems our technology solves better. Most managers in most companies will agree that their businesses are based on events. Does EDA help all businesses? Or only some of them? And if so, which ones?

Do our abstractions and methodologies help them? Do we have products that improve their lines of business? Do we have horizontal tools that can be tailored to their needs?

In fact we have all of the above, but we aren't clear about what we do have. And people in this room solve different aspects of problems related to events. And we aren't clear about what problems each of us works on. As a devil's advocate, this is my most compelling criticism.



Uniqueness of EDA on enterprise stack?

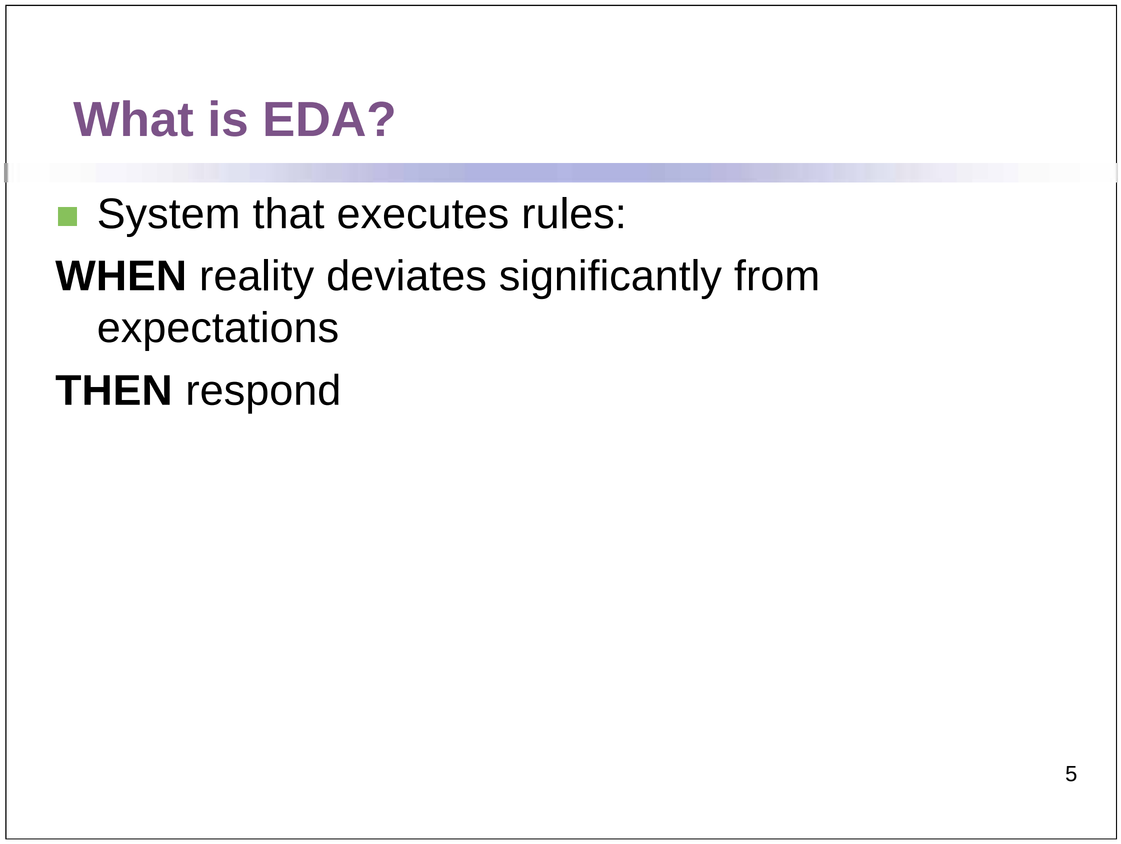
- DB: real-time DB
- Dashboard – BAM
- Messaging - ESB
- Rules Engines – real-time rules engines
- Business Intelligence – real-time BI
- Web Services
- …..
- Buy another tool, concept? Or reuse?

4

Almost all the features that our community claims to have is already a feature of products in the enterprise stack. Fast rule execution? That's in the new rules engines. Rapid business intelligence? That's in the new BI tools.

Explaining what we do better to our colleagues in our own companies and universities is difficult if we focus on new products or on performance. We need to start where Codd started with relational databases …. With ideas and algebra. Relational databases did not develop initially because it promised to improve DBMS performance by a factor of 10. It developed because of an abstraction --- a collection of elegant ideas. Performance did come soon, but it came later.

This is where David Luckham's book offers a signal service to the community. And Roy Schulte's speeches and articles.



What is EDA?

- System that executes rules:

WHEN reality deviates significantly from expectations

THEN respond

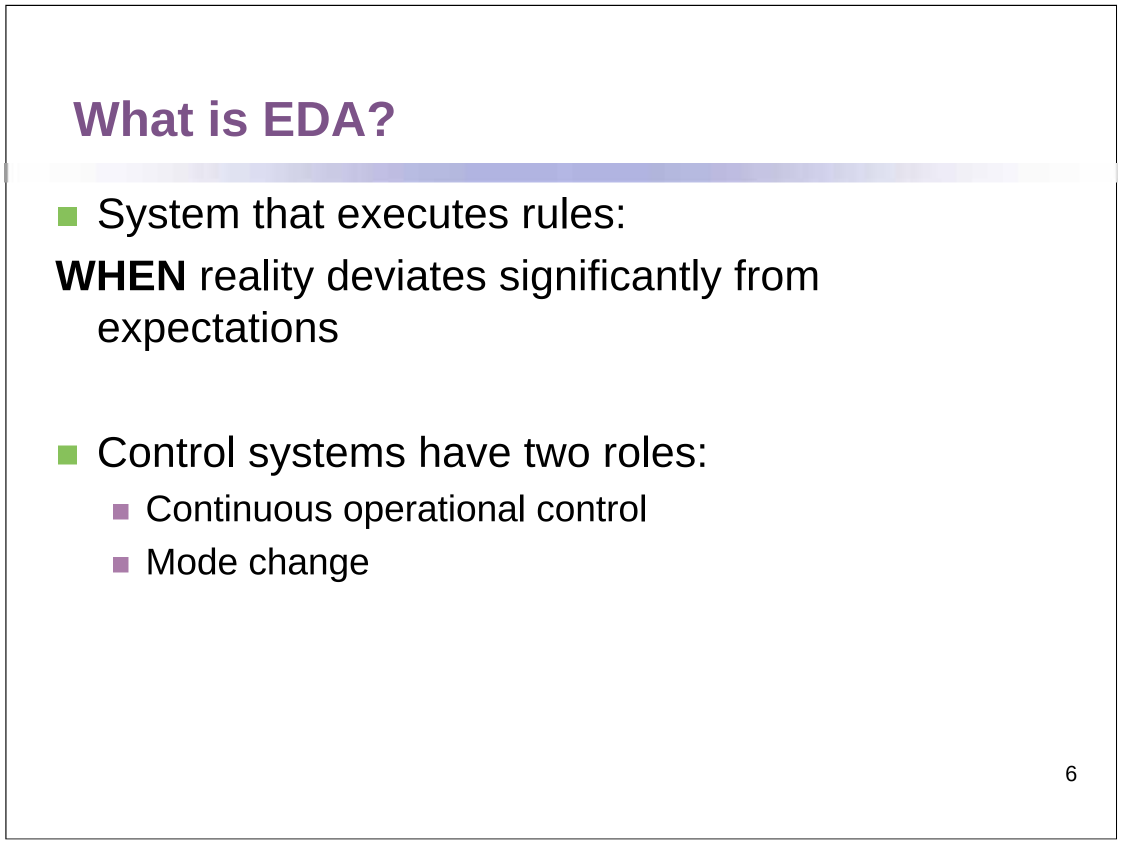
5

I will define an event-driven application as one that executes when-then rules. When reality deviates from expectations then respond. Let me repeat: the when clause deals with the deviation of reality from expectation.

The deeper levels of our brains are trained to ignore almost all the sensory information we receive. We **begin to pay attention** when reality, as determined by our sensory organs, is different from our model of reality. When we drive we have models of behaviors of other cars on the road; for the most part we ignore them because they behave as we expect them to. But, we start paying attention when the driver ahead of us suddenly slams on the brakes.

The change in the state of our world is that reality deviates from our expectation. The consequent change in our state is that we start paying attention. The first part of the then-clause is to pay attention or investigate further.

I'll come back to the important role of models in describing EDA because we tend to talk at cross purposes when we use different models. We also talk at cross purposes if we talk about different features of EDA. That's why I've started with this very specific feature …. The exogenous state change and the



What is EDA?

- System that executes rules:

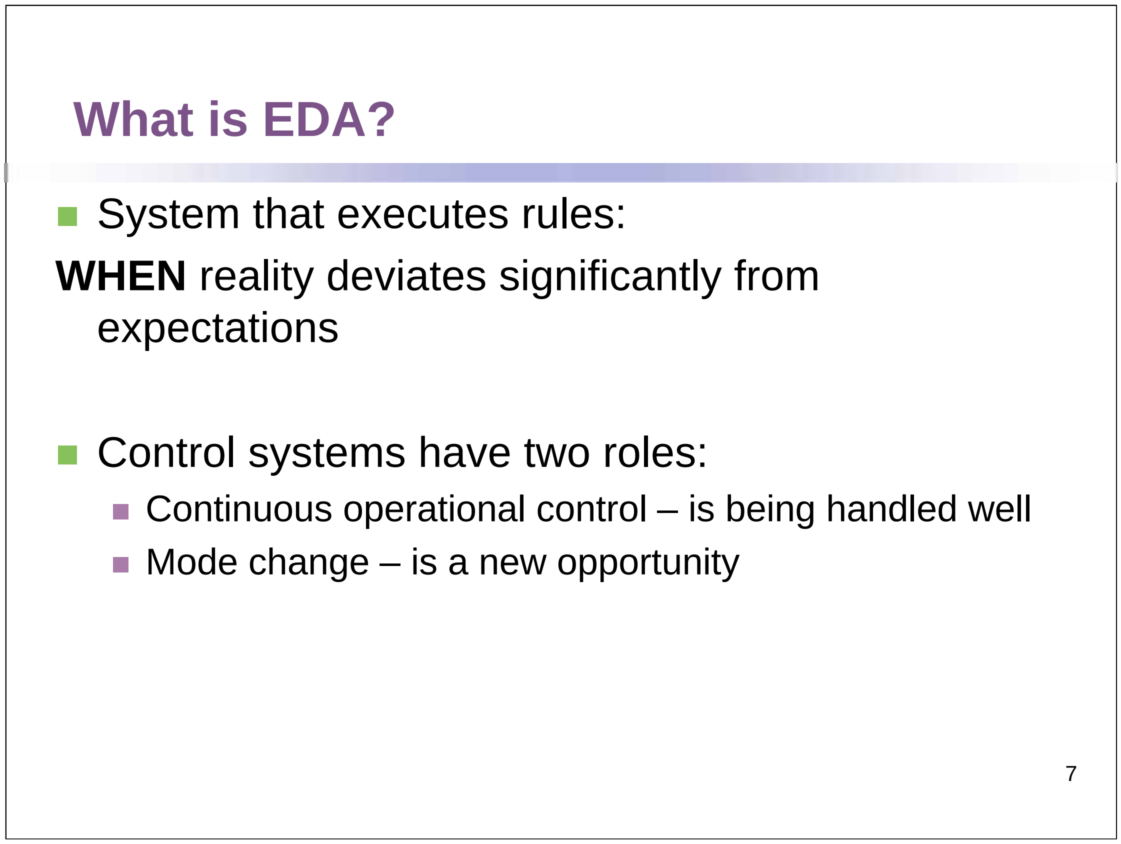
WHEN reality deviates significantly from expectations

- Control systems have two roles:
  - Continuous operational control
  - Mode change

6

An event-driven application can be defined as one that executes when-then rules. When an important situation occurs, then respond appropriately.

This definition generates lots of questions. Firstly, who specifies the rules? How many types of rules? Could rules be at cross purposes? How are when-clauses evaluated? What sorts of then-clauses are used in enterprises?



What is EDA?

- System that executes rules:

WHEN reality deviates significantly from expectations

- Control systems have two roles:
  - Continuous operational control – is being handled well
  - Mode change – is a new opportunity

7

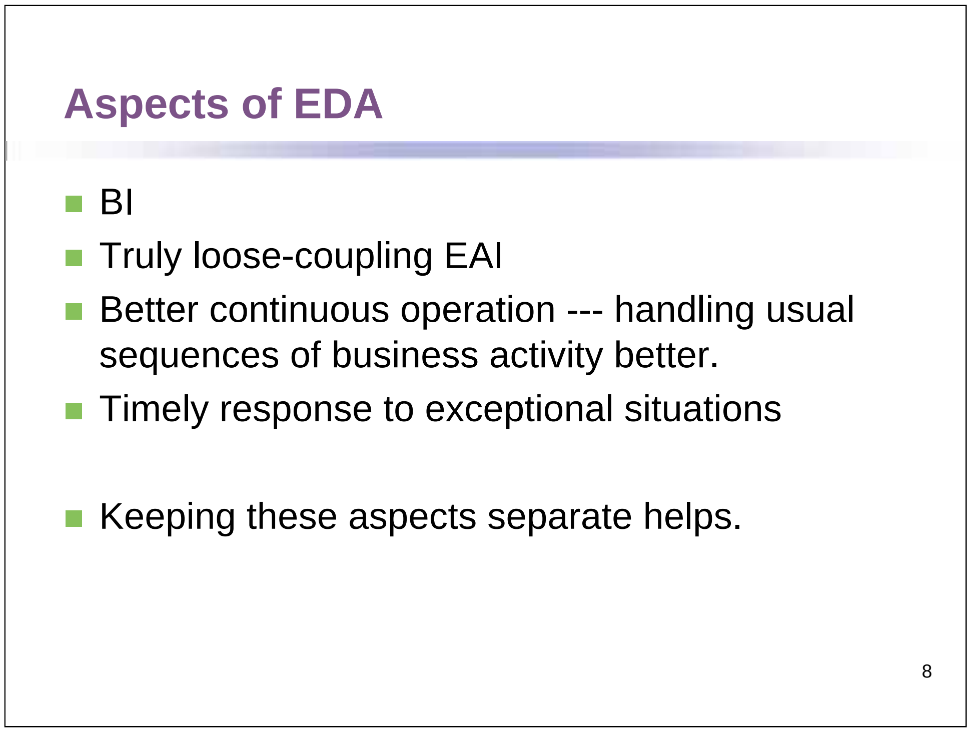
Control systems deal with events, control theory has been studied for 50 years. I have a lot to learn from control theorists.

Control theorists deal with two modes of operation. The normal mode is when the system is going according to plan. For example, when a plane is cruising according to its flight plan. A mode change occurs when the control laws have to be changed to deal with a new situation.

Playing the role of devil's advocate, I'll posit that problems of continuous operational control of business are already being solved by the existing enterprise stack or are already on roadmaps. The problem that remains to be solved is the detection and response to mode change.

Mode changes occur relatively rarely. Continuous operation happens all the time.

There are, of course, many many features to EDA including better continuous operation. But, unless we narrow our attention to specific limited features the value of EDA will remain fuzzy in our minds.



Aspects of EDA

- BI
- Truly loose-coupling EAI
- Better continuous operation --- handling usual sequences of business activity better.
- Timely response to exceptional situations

- Keeping these aspects separate helps.

8

There are at least 4 classes of related features of EDA.

1. Business intelligence gathered by analyzing repositories of events. Here events generated by business and IT processes are stored and analyzed. Valuable patterns can be detected from histories of events. Fraud, buying patterns, possible terrorist threats can be detected in this way.

2, We can think of events as the natural progression of increasingly loose coupling in EAI. SOA is a step on the way.

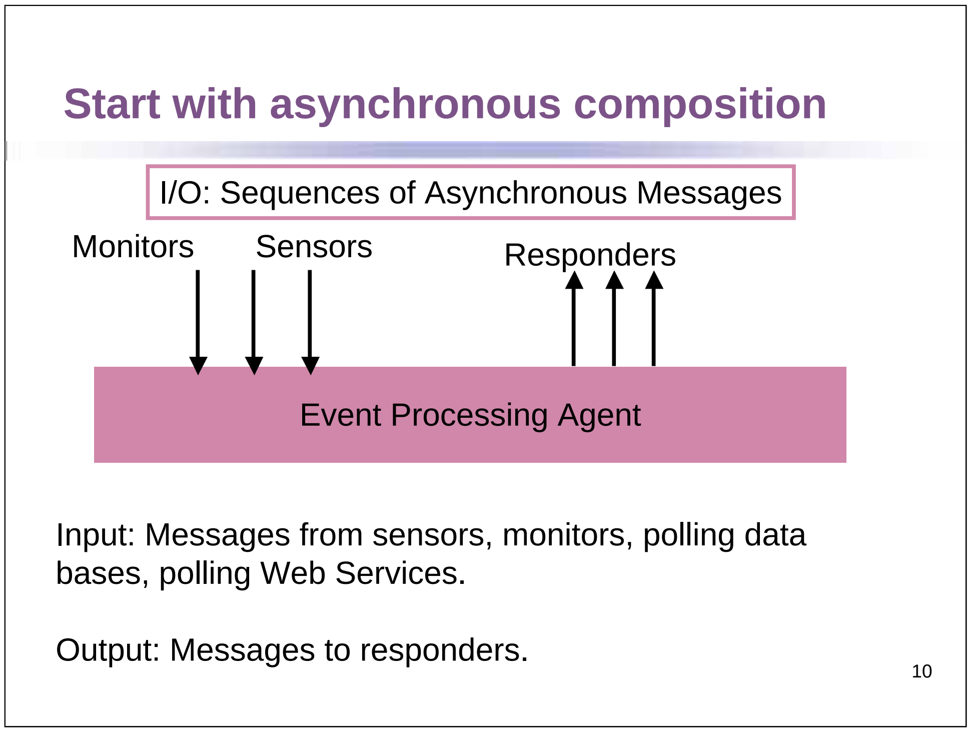
3. We could focus on better continuous operation.

4. Or we can focus on timely and appropriate response to exceptional situations.

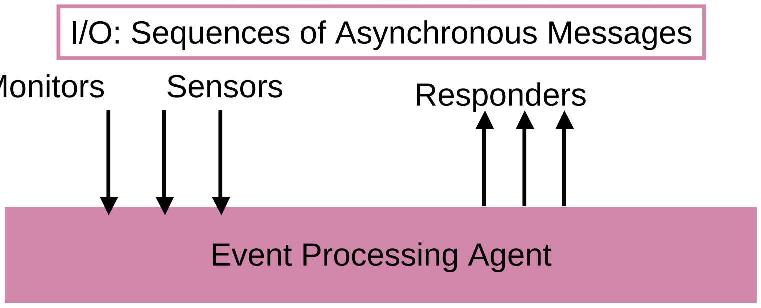
We are confusing these features in our discussions. True, all these features are a part of the *science* of EDA. But we need to keep these aspects apart when we talk about tools, programming languages and APIs.

Architecture:
Some thoughts

9



Start with asynchronous composition



Input: Messages from sensors, monitors, polling data bases, polling Web Services.

Output: Messages to responders.

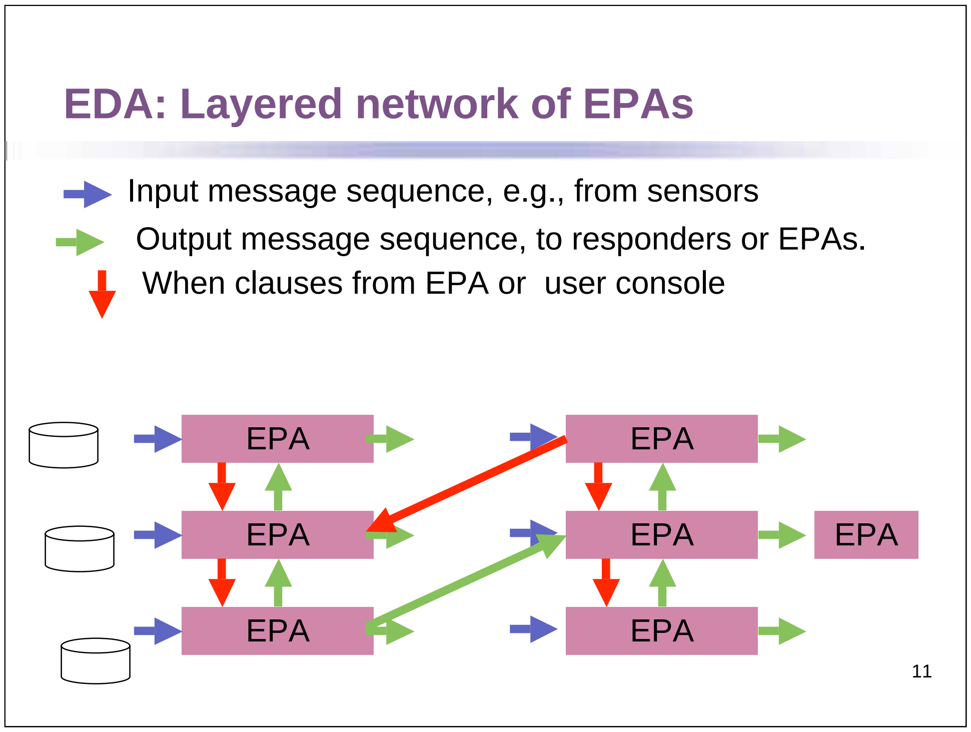
10

Let’s postulate that input and output consists of sequences of messages. These messages are asynchronous… not request-reply.

This part of the system does not deal with how to respond. That’s a business activity handled elsewhere.

Likewise, this part of the system does not deal with sensors or polling mechanisms to extract data from servers.

I/O is extremely important. Adapters to extract information are extremely important. But, these features are handled satisfactorily by other products in the enterprise stack.



EDA: Layered network of EPAs

- Input message sequence, e.g., from sensors
- Output message sequence, to responders or EPAs.
- When clauses from EPA or user console

An event-driven application, in my view, is a layered network of event-processing agents. The blue arrows indicate message sequences from sensors. The green arrows indicate message sequences to responders or to other EPAs in the same layer or higher layers. The red arrows indicate new when-then rules.

Green arrows go horizontally to responders that are not shown in this cartoon, or they go to other EPAs in higher layers.

Red arrows go down. Blue arrows are shown horizontally in this cartoon, but that's merely to make the slide less messy… the same sensor can feed EPAs at different levels.

This is not the most general structure. This structure is not an architecture at all; but it is a plea that any reference architecture, or functional architecture, be organized in layers.

The lowest layer in this picture deals with message streams with the highest throughput. The EPAs at this layer discard those messages that contain information that reality matches expectations. The EPAs pass on messages and other information when reality deviates from expectations. These messages, shown in green, can be passed to responders or EPAs at higher layers.

Filters used by EPAs at the lowest layer are simple. These simple models allow EPAs at this layer to handle large message rates. For example, a filter may be that the price of a barrel of oil increased by more than 2% in the last hour. Even such a simple filter can be thought of in terms of a model…. Expectation is that oil prices won't jump by 2% or more in an hour.

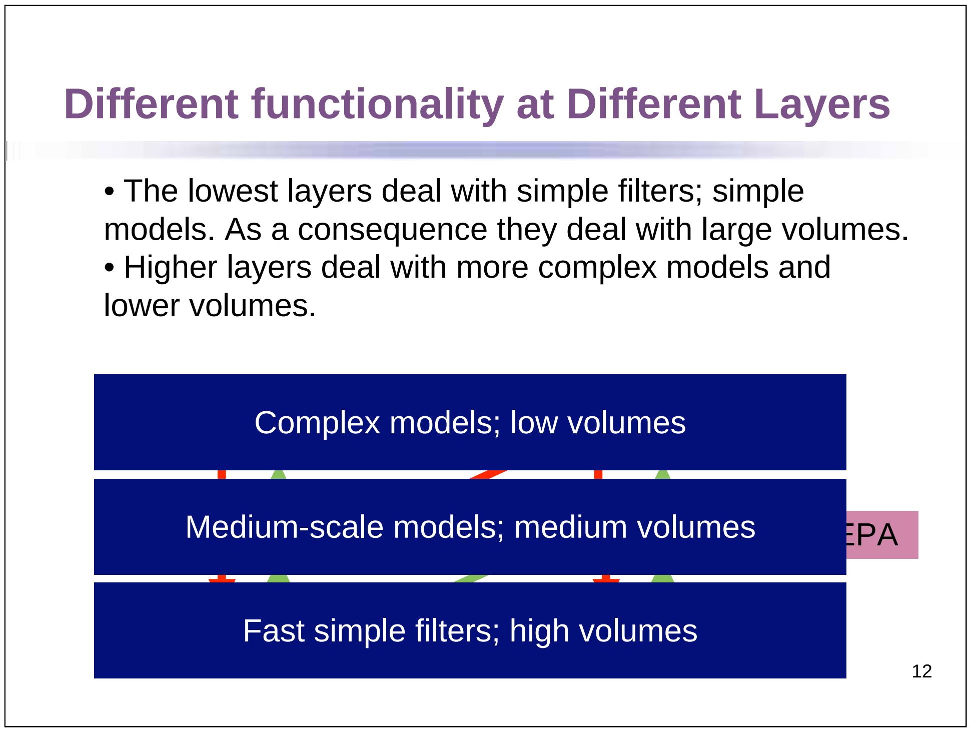
The rates at which messages are fed to EPAs higher up the stack is much lower. These EPAs deal with more complex models of reality. For example, a model of reality may be that price of gas, gold and oil change in a concerted fashion where the model is based on the current political situation.

An EPA at a higher layer in a stack may carry out optimizations and based on the optimizations it may change when-then rules in EPAs lower in the stack. This is shown by the red lines. For example, when a certain type of intrusion appears probable by using a model at a higher layer, then that EPA at the higher layer changes a when-then rule in a lower layer so as to deal with that kind of intrusion.

In effect, results of models at higher levels are used to set models or change parameters at lower levels.

I believe that this is the way that human beings, and indeed all animals, behave. A parent's ear may be especially tuned to the cry of a child who has had a bad day at school, while the parent may not hear a child who is happy and healthy. Our upper level model sets the parameter in our lower-level model (the ear).

Each layer is itself organized as a network of EPAs. For instance, in intrusion detection into a secure physical space, there may be a network of EPAs at the lowest layer, where each EPA is responsible for monitoring and filtering events in some geographic region. Likewise, an RFID application may have multiple EPAs at the lowest layer, where each



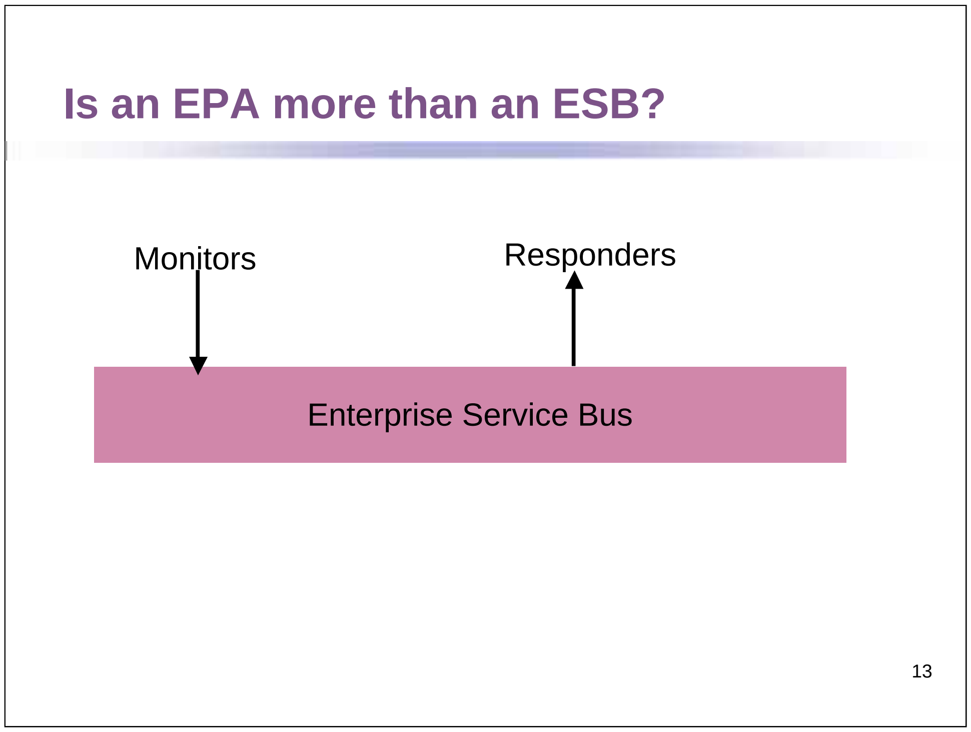
Different functionality at Different Layers

• The lowest layers deal with simple filters; simple models. As a consequence they deal with large volumes.
• Higher layers deal with more complex models and lower volumes.

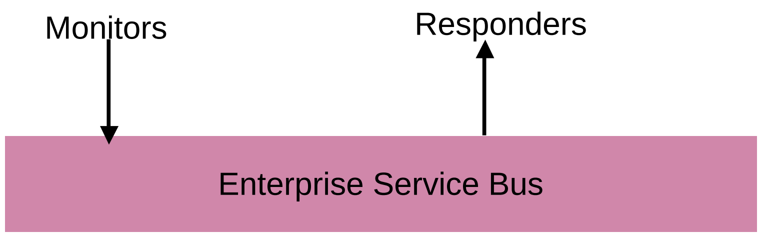
Complex models; low volumes
Medium-scale models; medium volumes
EPA
Fast simple filters; high volumes

12

The lowest layer deals with simple models and high volume



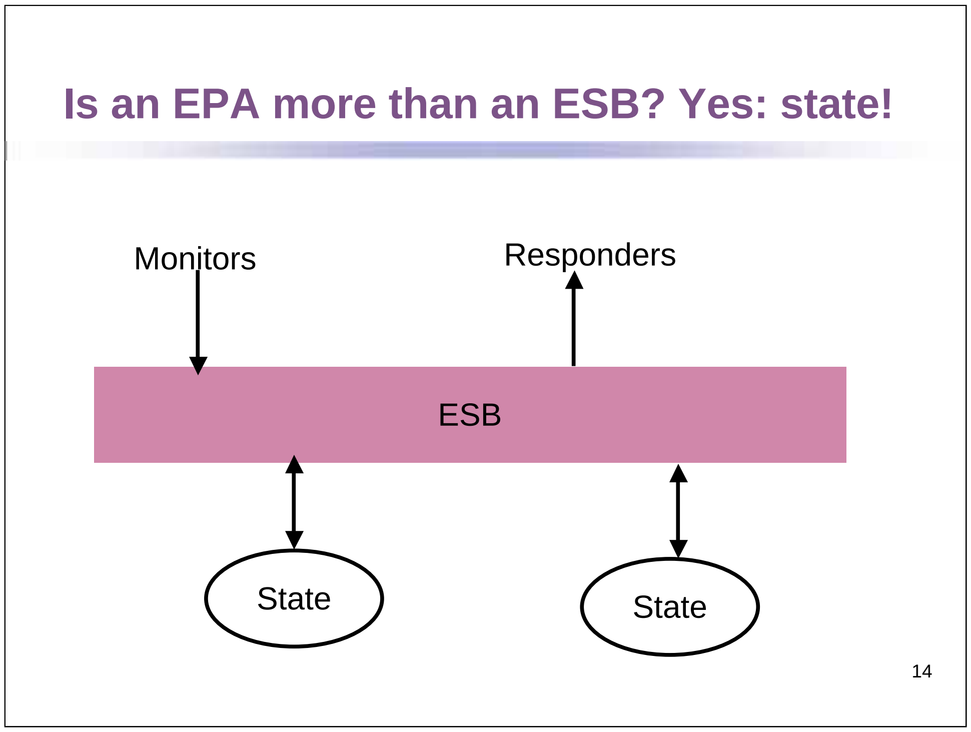
Is an EPA more than an ESB?



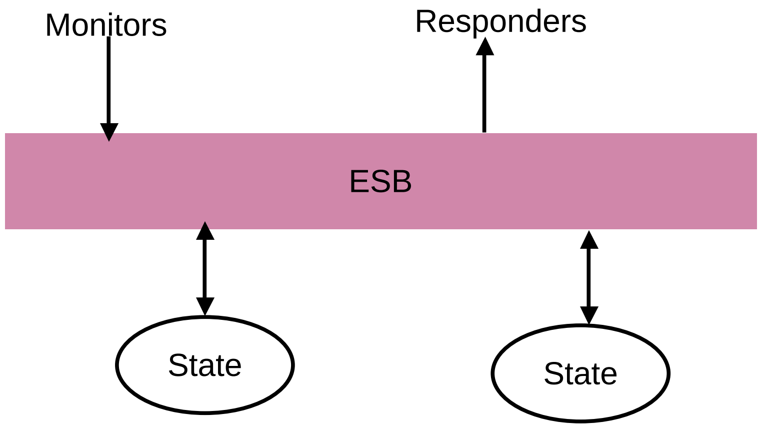
13

An enterprise service bus accepts messages, brokers messages, transforms messages, and allows users and software agents to subscribe for messages based on topic or content. So is an EPA an instance of an ESB?

We need to ask such questions of ourselves to understand – for ourselves – the incremental value that EDA brings to the enterprise stack.



Is an EPA more than an ESB? Yes: state!

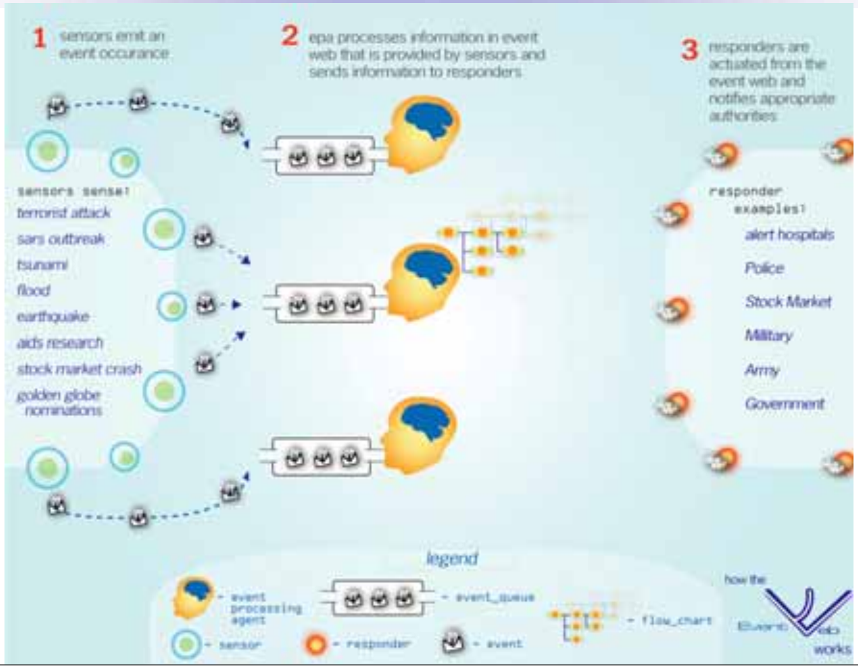


14

The key difference is state. Subscriptions in ESBs are subscriptions on messages. Subscriptions for EPAs are on state changes.

An agent in an ESB may subscribe for all messages with IBM in a field. An agent in an ESB cannot subscribe for notification of state changes such as a “buy IBM stock signal” based on trends in the last week.

EPA feature: Look OUTSIDE the enterprise



15

(This slide is from articles by Jonathan Lurie and me in the online magazine developer.com.)

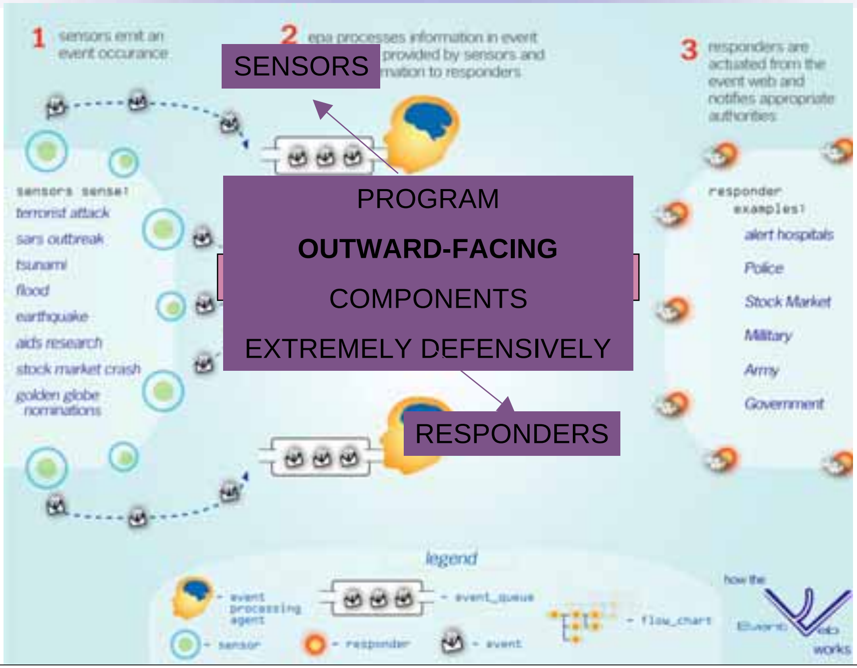
Enterprises are using information technology to look **OUTWARDS** at the world around them. Of course, enterprises already use IT to work with customers and suppliers. But, they are now also sensing a world around them that includes competitors, the government and academia. They are also getting feeds from an increasing collection of sensors.

State changes that happen outside the enterprise happen at times that are not controlled by the enterprise. Your competitor isn't going to offer your customer a better deal only on Tuesdays at 9AM.

Likewise, your competitor won't invoke a service on your machine to inform you each time your competitor changes prices.

SOA is ideal for certain kinds of activities within the enterprise. It is unsuitable for dealing with the larger world. That's where EDA is absolutely necessary.

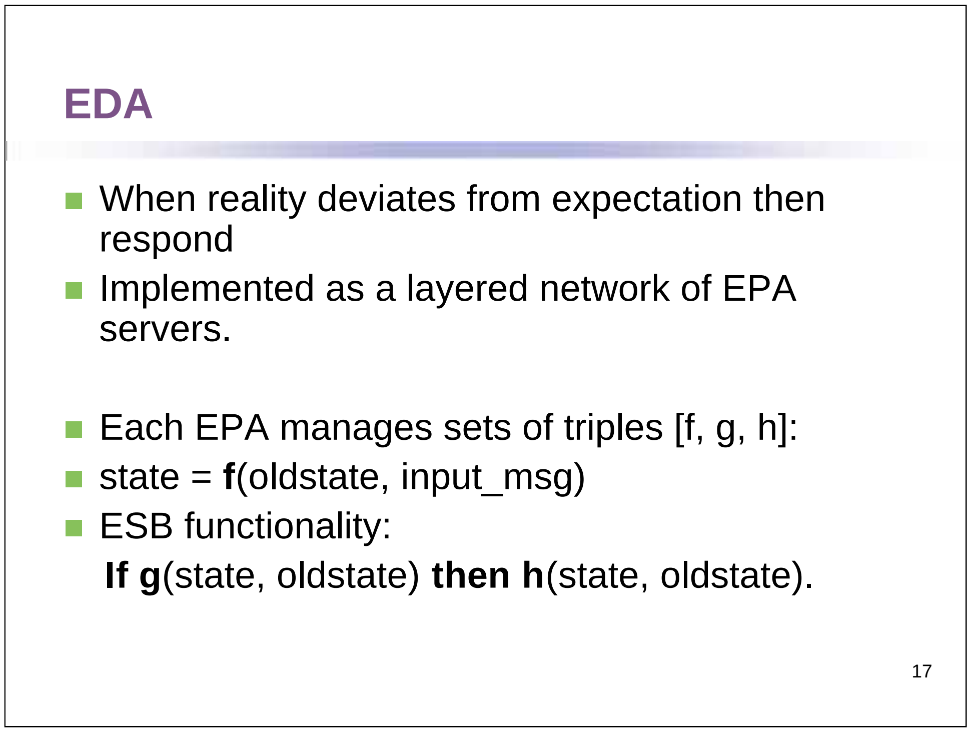
Defensive Programs: Sensors & Responders



16

The “outward-facing” components of an event-driven application are programmed extremely defensively. The sensing portion may have to poll the environment rather than have data pushed to it.

Outward-facing enterprises have to have some aspect of EDA because they have to deal with asynchrony and external state changes. You can’t react appropriately to external state changes unless you have some state yourself.



EDA

- When reality deviates from expectation then respond
- Implemented as a layered network of EPA servers.

- Each EPA manages sets of triples [f, g, h]:
- state = **f**(oldstate, input_msg)
- ESB functionality:

  If g(state, oldstate) **then h**(state, oldstate).

17

I'd like to remind ourselves of a couple of points: Let's focus on the feature: when reality deviates from expectation then respond.

And the architecture consists of a layered network of EPAs.

Each EPA, or more accurately EPA server, manages a set of rules. Each rule is a triple of functions f, g, h.

F is the state-update function. It is triggered when a message arrives.

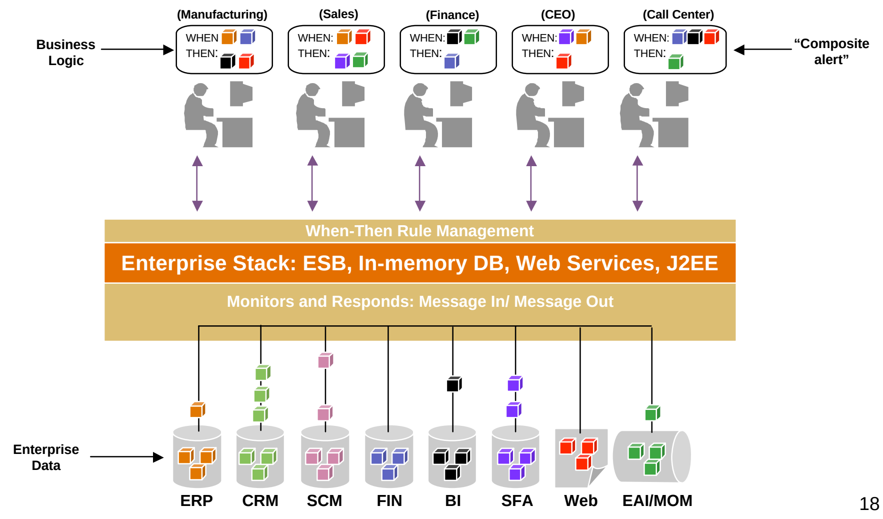
G is a guard ---- it determines whether a state change should trigger a response.

H is an action --- the response to be taken if the state-change satisfies the guard.

Of course, we can wrap all three functions into a single one. But, keeping the three aspects: state-change, guard on state change, and reaction to state change separate helps.

Each user may have many such triples. Each EPA may have many users. Each of the three parts --- f, g, h ---- will be common across the rules within an EPA. Separating the three parts makes the commonality among different rules for each of the three parts more evident.

An EDA Vision

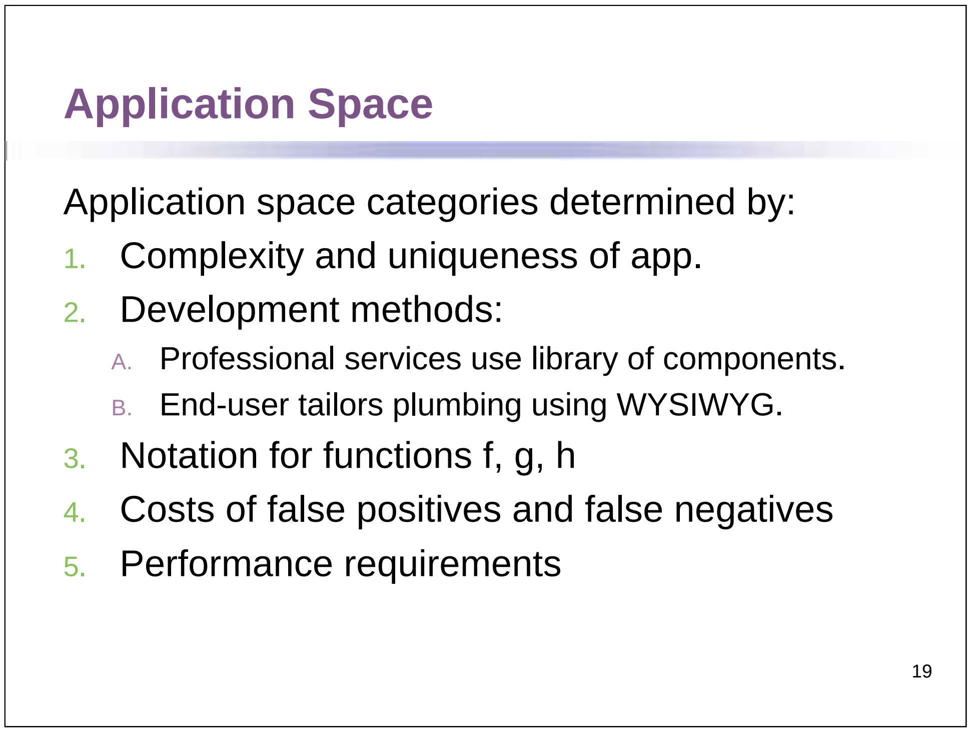


18

A vision of EDA is that it is a thin middle layer that wraps IT within an enterprise so as to allows manager to use their skills where it really matters ---- when reality deviates from expectations.

Aspects of continuous operations may be outsourced to contractors. Dealing with strategy and exceptions cannot.

If this is our view of EDA then it colors the discussion we're having today. There are other views that may be more important and more relevant. But, unless we make our views clear we'll be talking at cross purposes. And we won't be able to identify the repeatable asset.



Application Space

Application space categories determined by:

1. Complexity and uniqueness of app.
2. Development methods:
   A. Professional services use library of components.
   B. End-user tailors plumbing using WYSIWYG.
3. Notation for functions f, g, h
4. Costs of false positives and false negatives
5. Performance requirements

19

The application space for EDA, in the context of the last picture depends on these features.

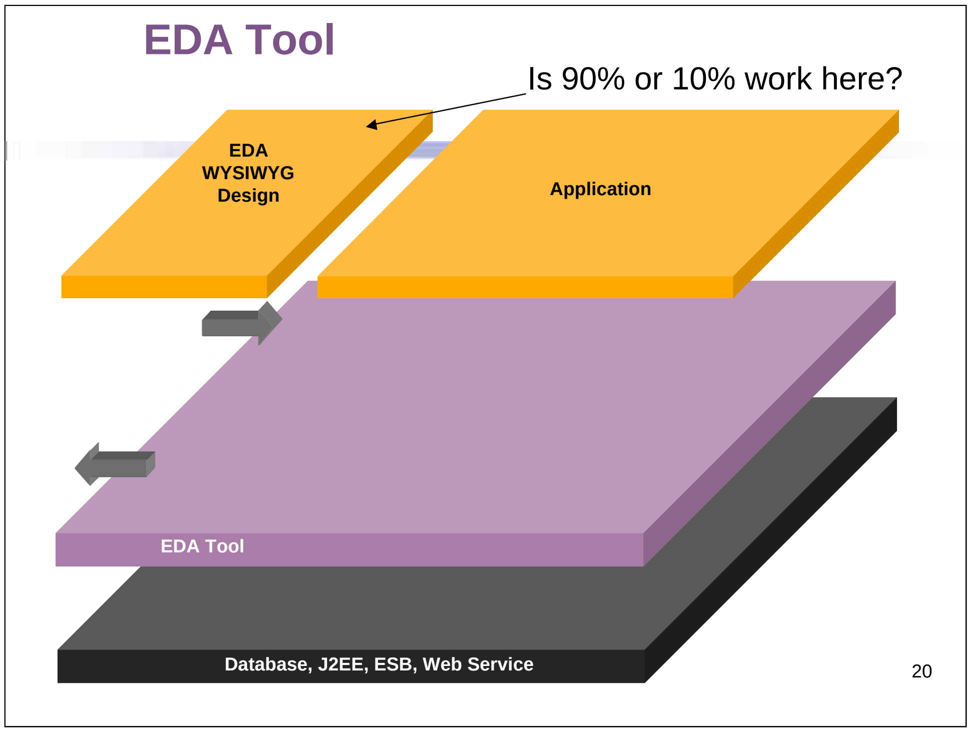
How complex and unique is the overall app? If EDA is transforming different aspects of the enterprise --- the supply chain, or CRM --- how different are these applications different from each other? Do all supply chain applications have greater similarity than all EDA applications? This is the age-old question of vertical versus horizontal, of lines of business on the one hand or IT functionality on the other.

An application can be developed by professional services using libraries of components and the existing enterprise stack. Or an app can be developed by end-users tailoring plumbing by using WYSIWYG tools. What makes sense? That depends on your view of which problem we're solving?

A critical aspect of this last question is the notation for f,g,h: state-change, guard on state-change and response to state-change. For professional services Java or .NET class libraries that implement f, g, h are ideal. For IT within an enterprise a WYSIWYG graphical tool may be better. For a business user a business-specific UI is necessary. When we talk about notations ---- extensions to SQL or XPath --- that's a useful discussion, but we must frame this discussion in the context of which category of user we're trying to help.

It's all about the total cost of ownership, and a key part of that is the cost of false positives --- responding to situations that don't need responses. The huge costs of false negatives is accepted. But we tend to overestimate the ratio of costs of false positives to false negatives. Systems with too many false positives are ignored.

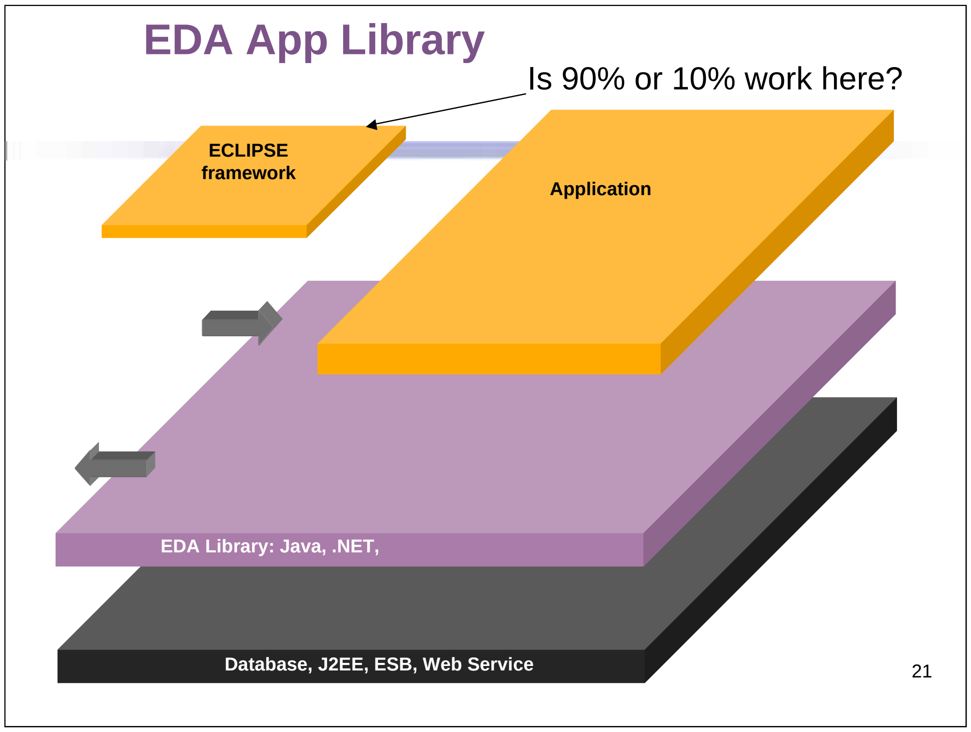
The kind of tool and methodology we use depends critically on performance requirements.



EDA Tool

One view of an EDA plumbing tool is that a WYSIWYG wraps enterprise IT to give continuous operational control and exception-management control. In this view, a what-you-see-is-what-you-get GUI tool is used to wrap a horizontal platform to get any desired collection of vertical applications. To take an extreme and unfair point of view, the idea here is that EDA plumbing with a designer console can convert an enterprise to a real-time enterprise.

Of course, no vendor claims this. But some times, perhaps merely by accident, our talks and whitepapers suggest that we have this capability.

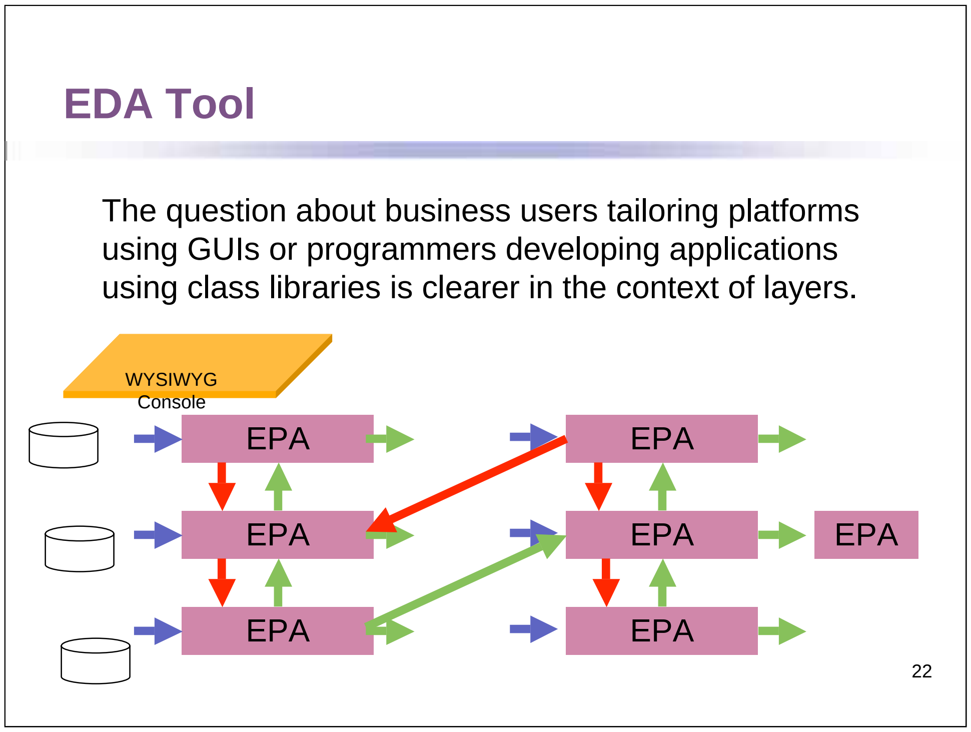


EDA App Library

A different view, is that this feature is done through libraries by professional services. Here the value proposition is that professional services will take 18 person-years rather than 20 person-years to implement an application.

Which view is correct? They are both correct. It depends on which part of the layers of EPA we are dealing with.

Unless we separate the different layers of EPAs our conversation here will be fuzzy.



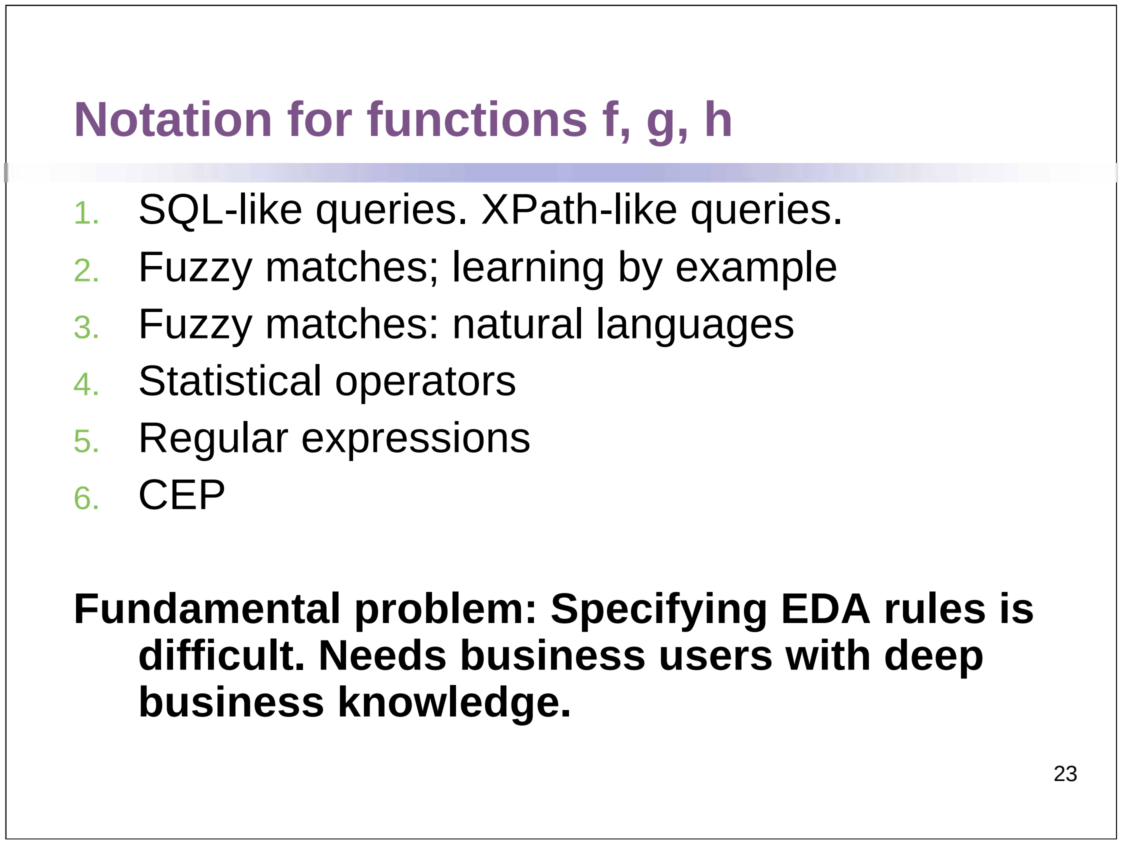
EDA Tool

The question about business users tailoring platforms using GUIs or programmers developing applications using class libraries is clearer in the context of layers.

WYSIWYG Console
EPA
EPA
EPA
EPA
EPA
EPA
EPA
22

We don’t expect a single graphical user interface to serve in tailoring all the layers to create all business-specific event-driven applications.

So, we need to be clear about what layers each GUI tool is designed for, and what class of user.



Notation for functions f, g, h

1. SQL-like queries. XPath-like queries.
2. Fuzzy matches; learning by example
3. Fuzzy matches: natural languages
4. Statistical operators
5. Regular expressions
6. CEP

Fundamental problem: Specifying EDA rules is difficult. Needs business users with deep business knowledge.

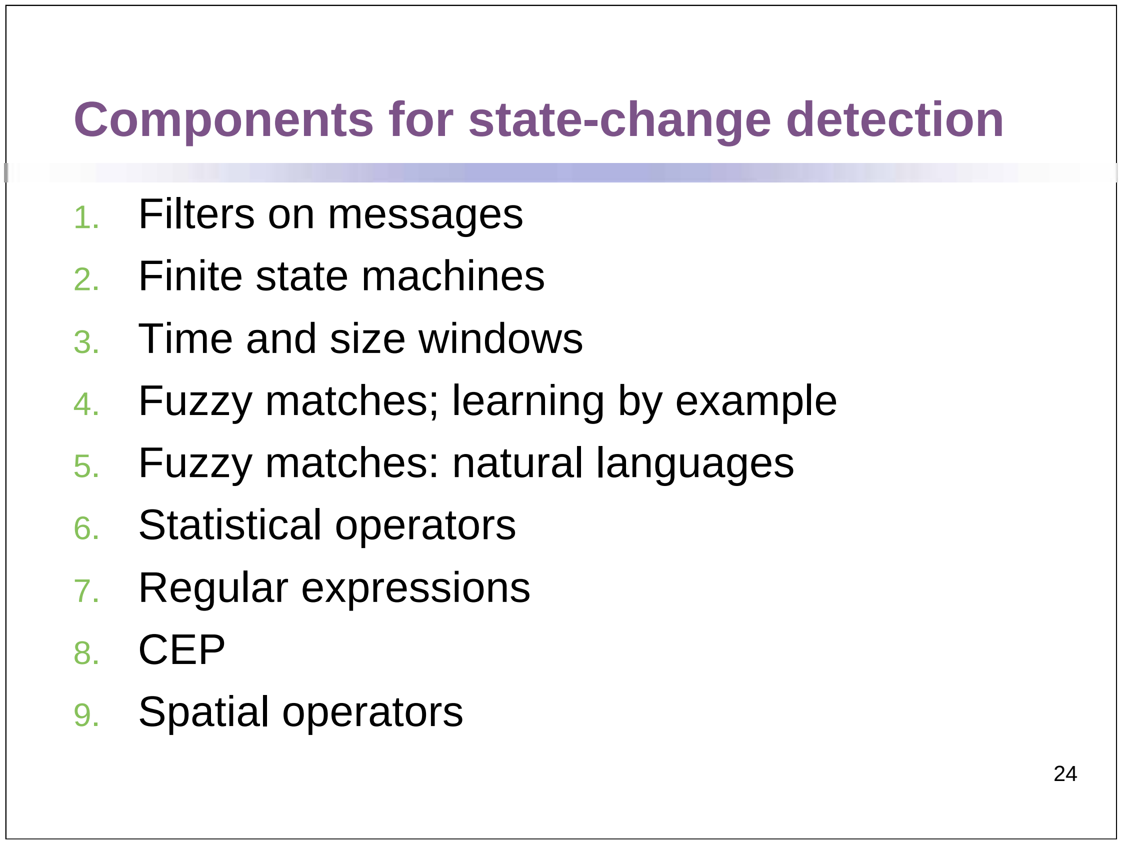
23

Now let’s look at another aspect of the incremental cost: the development of components needed for EDA and the effort in integrating these components to provide EDA functionality.

There are many notations for f, g, and h. Many apps, especially externally-facing apps, have to deal with fuzzy matches such as matches on partially-structured text documents, or images.

It’s premature to talk about a single Event Processing Language standard until we find out what types of f, g, h functions are relevant for a given application.

We probably need different EPLs at different layers of the EPA network.



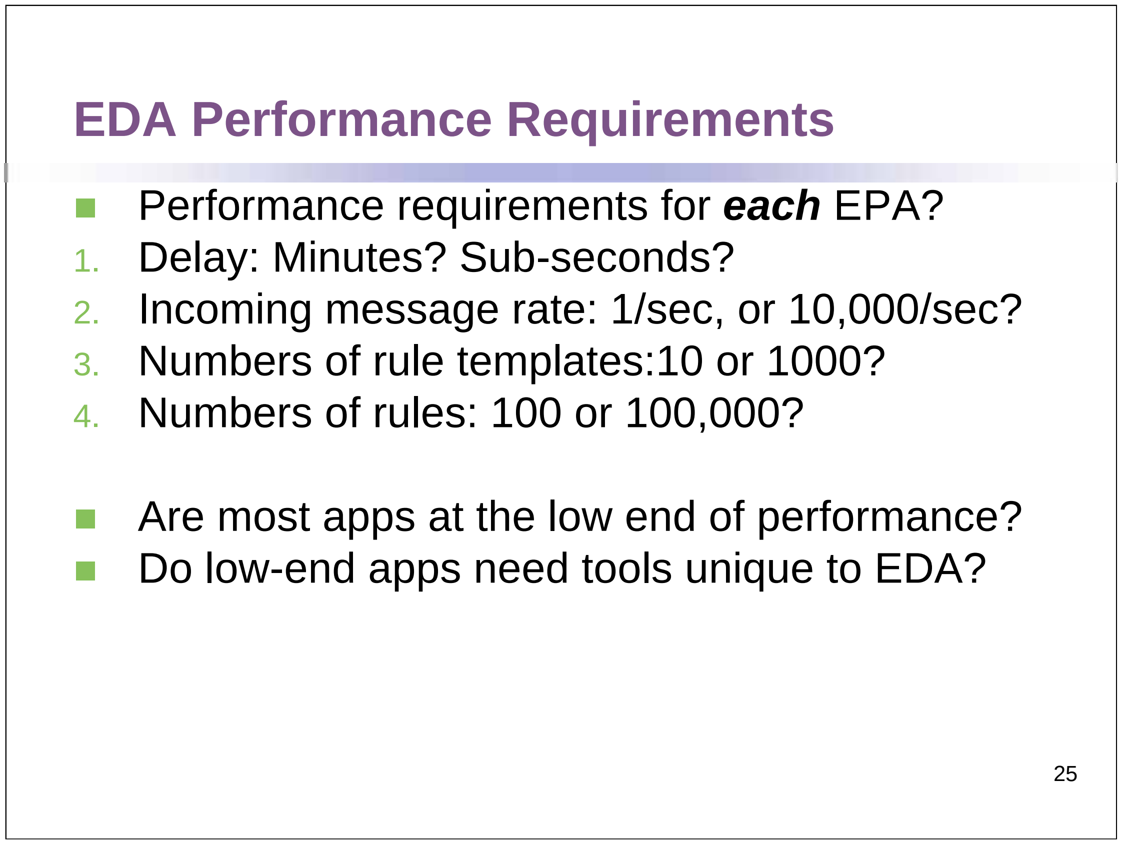
Components for state-change detection

1. Filters on messages
2. Finite state machines
3. Time and size windows
4. Fuzzy matches; learning by example
5. Fuzzy matches: natural languages
6. Statistical operators
7. Regular expressions
8. CEP
9. Spatial operators

24

Since the state-change, guard on state-change, and reaction to state-change functions are very different for the different types of matches – such as fuzzy matches and complex statistical matches --- I don't expect a single event processing language to fit the entire space. An alternative view is to design the system as a composition of components that deal with different aspects of state-change recognition. Output from a filter that only passes through stocks for which some statistics hold, and output from a component analyzing news stories, may be fed to a component that analyzes whether a stock is likely to be volatile in the immediate future.

I suggest we work at the level of *desired functionality*, and compositional structures of components that provide this functionality before we discuss standards for EPLs.



EDA Performance Requirements

- Performance requirements for ***each*** EPA?

1. Delay: Minutes? Sub-seconds?
2. Incoming message rate: 1/sec, or 10,000/sec?
3. Numbers of rule templates:10 or 1000?
4. Numbers of rules: 100 or 100,000?

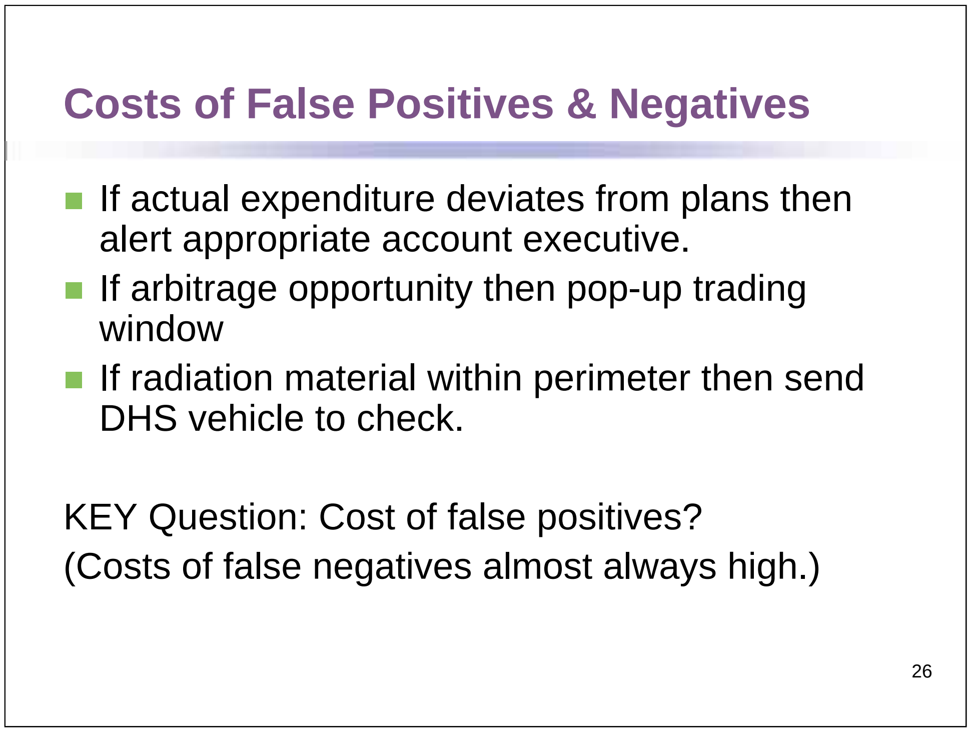
- Are most apps at the low end of performance?
- Do low-end apps need tools unique to EDA?

25

All enterprises generate events rapidly. Consider telephone calls to a customer service center. If we treat each call initiation and termination as events, the service center itself will generate huge rates of events. If we take an inventory of all the events in an enterprise rates of 10,000 to 100,000 per second are not unusual. But these are rates are not necessarily the key constraints for designing EPAs.

An EDA consists of layers of EPAs, and only the lowest layers with simple models deal with high volumes. Secondly each layer is a network of EPAs each of which deals with a different geographic region or some part of the overall problem. The load on each EPA can be made manageable by designing the EPA layer and the EPA network within each layer.

In our community we often lead with performance issues. But, performance limits of each separate EPA are only concerns in a small number of niche applications. These applications are hugely profitable. But, we can have more impact on society if we start with concepts and not performance.



Costs of False Positives & Negatives

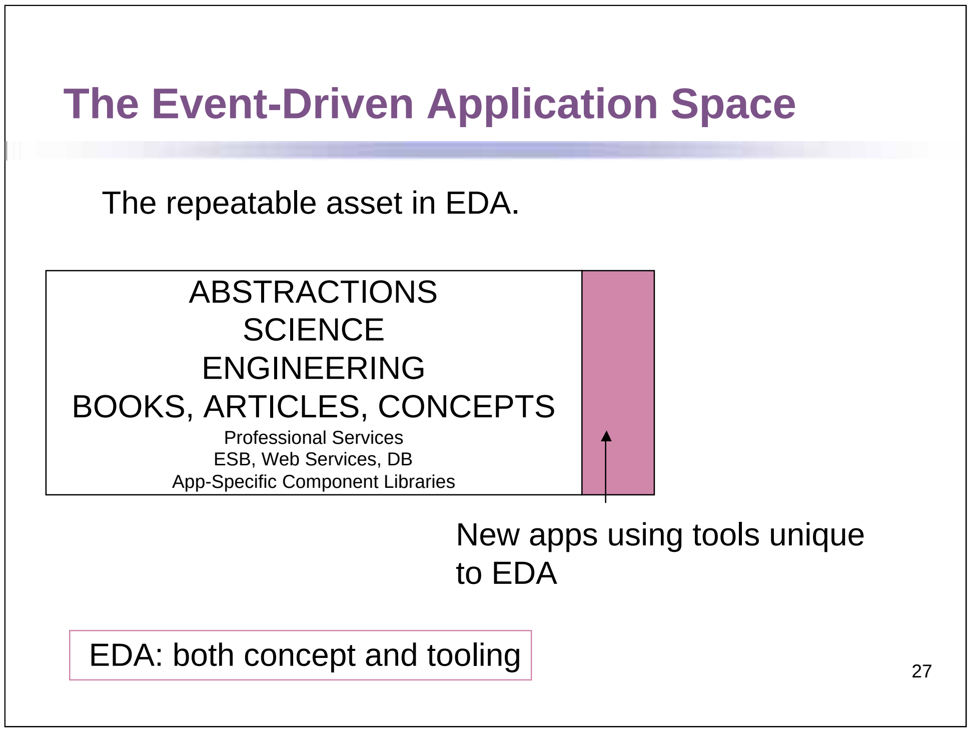
- If actual expenditure deviates from plans then alert appropriate account executive.
- If arbitrage opportunity then pop-up trading window
- If radiation material within perimeter then send DHS vehicle to check.

KEY Question: Cost of false positives?

(Costs of false negatives almost always high.)

26

Here are some examples of the costs of false positives and negatives. The costs vary a great deal. The costs of false negatives are high in most apps. We underestimate the costs of false positives because a false positive often takes the form of an alert to a human being, and we don’t value that person’s time. So, the alerts are ignored.



The Event-Driven Application Space

The repeatable asset in EDA.

ABSTRACTIONS
SCIENCE
ENGINEERING
BOOKS, ARTICLES, CONCEPTS
Professional Services
ESB, Web Services, DB
App-Specific Component Libraries
New apps using tools unique to EDA

EDA: both concept and tooling

27

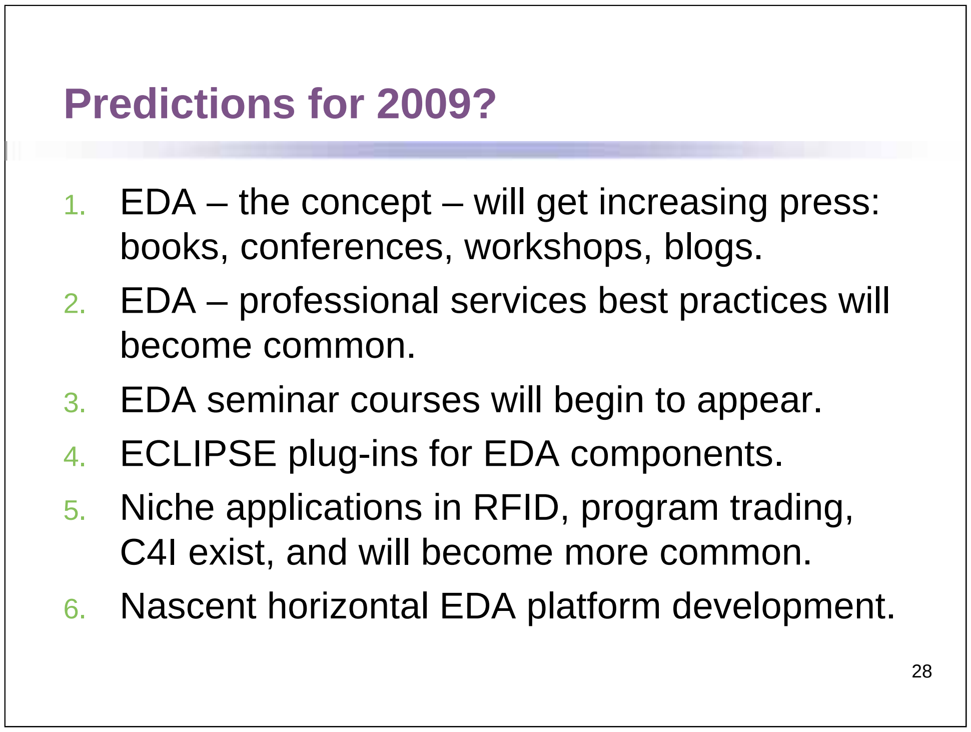
Most enterprises are event driven. The science of EDA can have a huge impact on society because most enterprises are event driven.

How can we have the most impact? By focusing on the science of EDA. The abstractions. The concepts. Design methods. How best to maintain event systems.

There are 5 niche applications that are heavily event-centered. Program trading is one of them. Companies in these niche spaces have developed and used event-driven applications for years. We can impact these niche applications by providing them with new tools unique to EDA.

But there are 995 applications that can benefit from the emerging science of events even if they don’t buy new EDA tools.

Vendors naturally focus on the niche verticals. But as academics, and as a community, we should perhaps pay more attention to the 995 verticals where relatively little money will be spent on new EDA tools.



Predictions for 2009?

1. EDA – the concept – will get increasing press: books, conferences, workshops, blogs.
2. EDA – professional services best practices will become common.
3. EDA seminar courses will begin to appear.
4. ECLIPSE plug-ins for EDA components.
5. Niche applications in RFID, program trading, C4I exist, and will become more common.
6. Nascent horizontal EDA platform development.

28

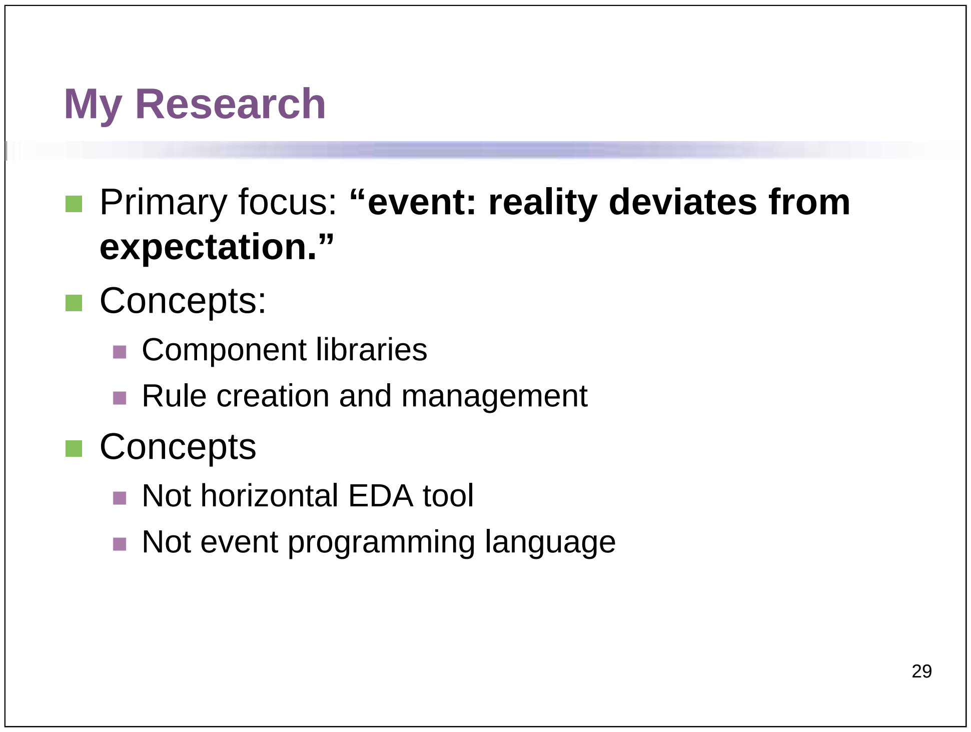
We've been asked to give predictions. Here are some.

The concept of EDA, the abstraction, the science,… will grow organically and due to our efforts as a community.

Best practices for this space will get codified. Java and .NET libraries will begin to appear.

Applications in the niche verticals will become common. These niche verticals are important and also deal with huge amounts of money.

Horizontal EDA platforms exist at the messaging and database layer in the enterprise stack in the form of ESBs with greater functionality, faster rules engines, faster BI. New horizontal EDA platforms will begin to become integrated with the enterprise stack and start to become more widely used.


My Research

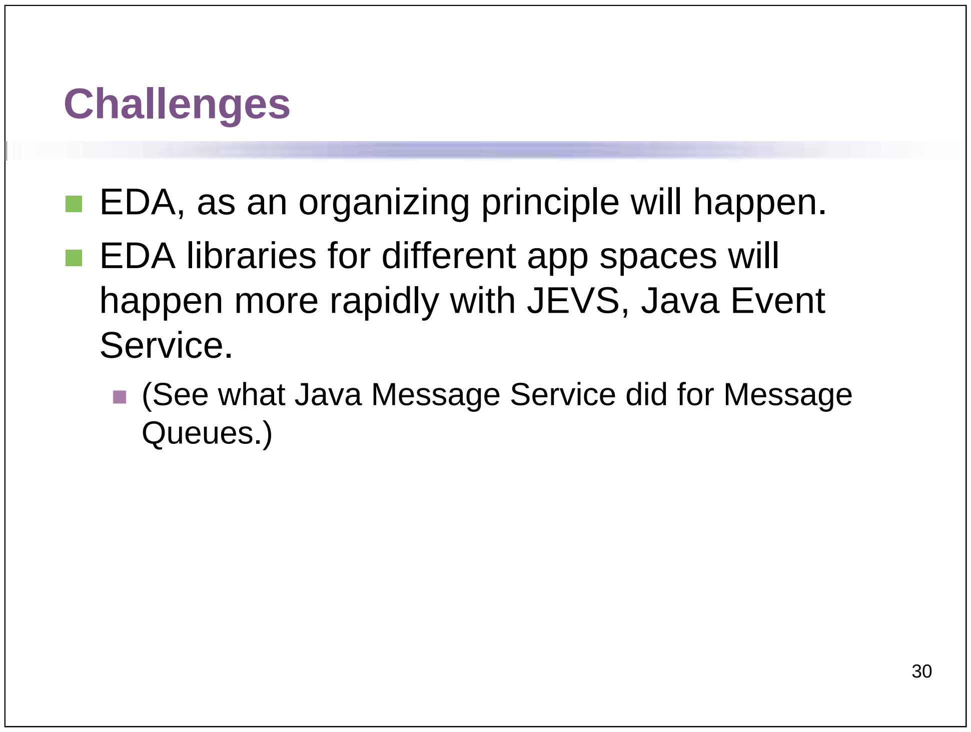
- Primary focus: **“event: reality deviates from expectation.”**
- Concepts:
  - Component libraries
  - Rule creation and management
- Concepts
  - Not horizontal EDA tool
  - Not event programming language

29

We were asked to talk about our research. Mine starts with the thesis that the events that matter are when reality deviates from expectations. We specify expectations by models of reality. How are models specified? How can models be checked against reality each time a message arrives, or if a message doesn’t arrive?

I am working on incremental algorithms for detecting significant state changes, and incremental algorithms for optimization… incremental because these algorithms make incremental changes each time a single message arrives.

The work we do at Caltech is at the level of components that can be composed easily. It is not at the level of programming languages or a horizontal EDA tool.



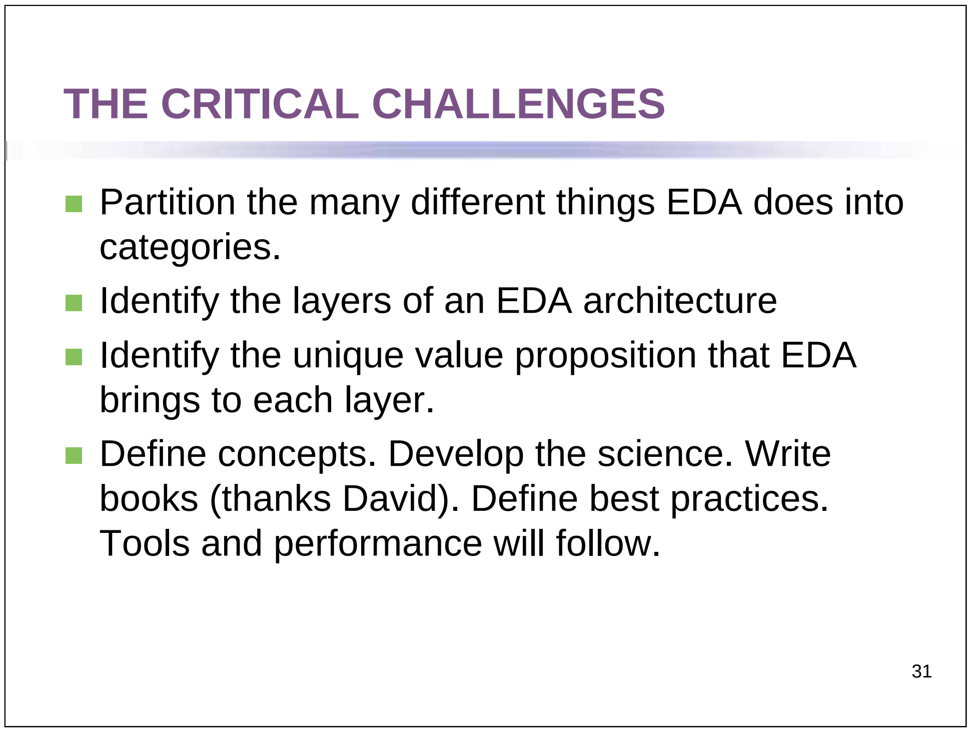
Challenges

- EDA, as an organizing principle will happen.
- EDA libraries for different app spaces will happen more rapidly with JEVS, Java Event Service.
  - (See what Java Message Service did for Message Queues.)

30

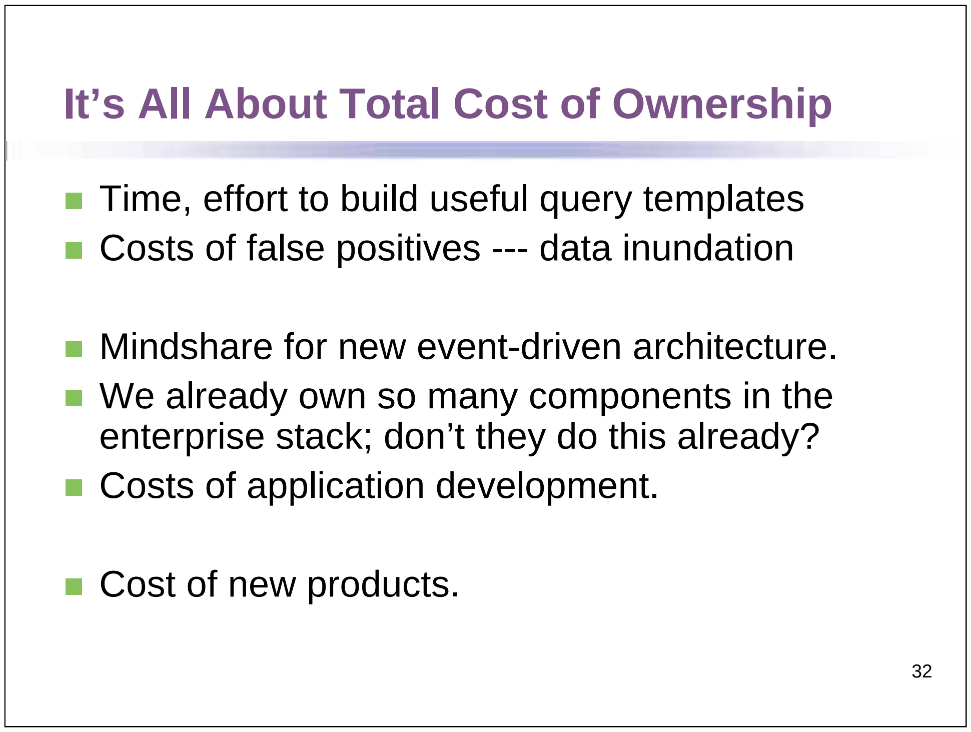
I think EDA as a science will develop.

I think libraries of components that help solve problems in the event space will be developed by industry and academia.



THE CRITICAL CHALLENGES

- Partition the many different things EDA does into categories.
- Identify the layers of an EDA architecture
- Identify the unique value proposition that EDA brings to each layer.
- Define concepts. Develop the science. Write books (thanks David). Define best practices. Tools and performance will follow.



It’s All About Total Cost of Ownership

- Time, effort to build useful query templates
- Costs of false positives --- data inundation

- Mindshare for new event-driven architecture.
- We already own so many components in the enterprise stack; don’t they do this already?
- Costs of application development.

- Cost of new products.

32

I’d partition costs into three categories. Costs specific to EDA; costs dealing with any significant change in the enterprise software stack, and costs of new products.

A huge effort is required from business users to develop an effective EDA app. Useful when-then rules require deep business knowledge. The cost of getting a trading hot-shot to spend a week developing and testing templates for program trading is substantial. These costs may be more important than the costs of the tool.

The costs of false positives should be considered at the outset. You can only reduce false positives by developing rules that make sense for the end business user. And this requires the business user’s time. If false positives are a little too high, business users will stop paying attention and the whole effort is wasted.

The second category deals with change in the enterprise stack. IT in the enterprise is still digesting SOA. Now we want mindshare for something new. And the value of SOA is just showing up for the enterprise. Secondly, the major companies already have products that offer many of the features that EDA claims to offer. The unique value offered by EDA isn’t clear yet.

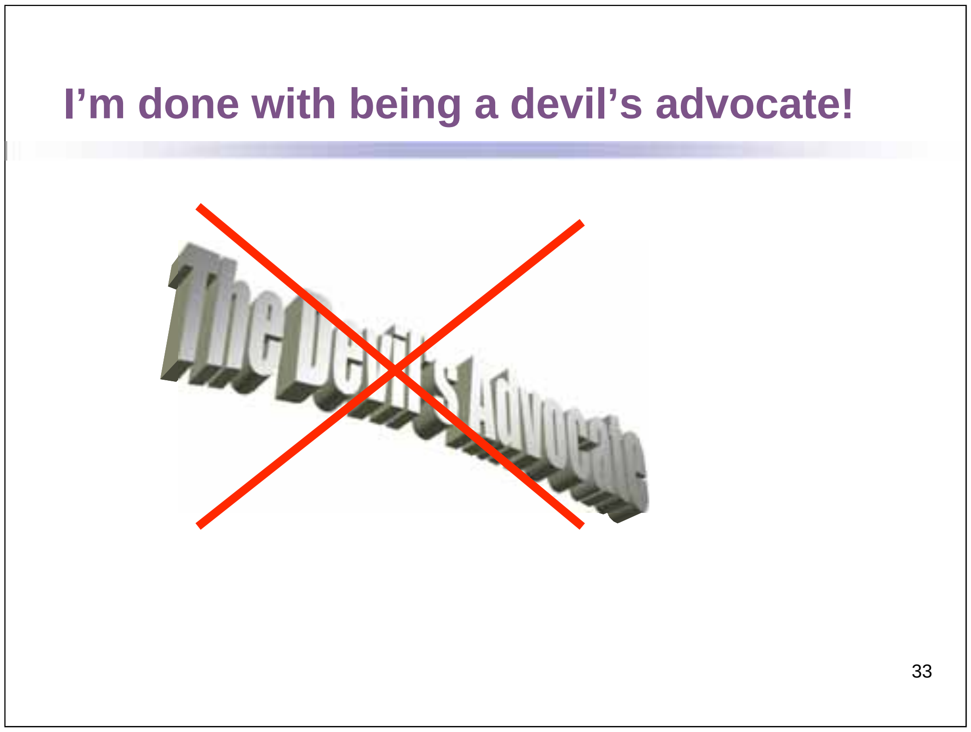
The cost of developing the final application that benefits the end user is substantial. This cost will decrease as professional services organizations get libraries and best practices for different verticals. But at this point the cost of an EDA app is substantial.

The third category deals with costs of products. That’s the least important part.

Industry and academia are working on the first category.

The second category requires education and time. I don’t see a way to speed that up.

The third category is less important than the first two.



I'm done with being a devil's advocate!

33

The military uses red teams and blue teams to evaluate strategy. The church uses devil's advocates. These roles are uncomfortable, but they serve a purpose.

I believe that EDA, ESP, CEP,… has a great deal to offer all enterprises. My role as devil's advocate is point out some of the challenges in getting to where we want to be and where we will be.