The Four Levels of Requirements Engineering for and in Dynamic Adaptive Systems

Daniel M. Berry, U Waterloo
Betty H.C. Cheng, Michigan State U
Ji Zhang, Michigan State U

Abbreviations

DAS	**= Dynamic Adaptive System**
RE	**= Requirements Engineering**
AR	**= Adapt-Ready (Adaptation-Ready)**

Dynamic Adaptive Systems

A DAS is a computer-based system (CBS) that

- **is capable of recognizing that the domain with which it shares an interface has changed and**
- **is capable of changing its behavior to adapt to the changing conditions.**

DASs

A lot of work is being done to develop technology to support DASs.

Interest in DASs is motivated by increasing demand for pervasive and mobile computing.

Motivation for Levels

We noticed that

1. **RE is always *about* input and a system’s response to it.**

   That is, RE determines

   - **the kinds of input a system may be presented and**
   - **the system’s responses to these inputs.**

Motivation Cont'd

2. **A DAS, *S_{AR}* is *doing* RE at run time!**

   That is, *S_{AR}* is determining *as it is executing*

   - **the kinds of input *S_{AR}* may be presented and**
   - ***S_{AR}*'s responses to these inputs.**

Motivation Cont'd

But that's not the *only* RE done about S_{AR}.

Humans are doing lots of RE about S_{AR} and about S_{AR}'s own RE!

So we thought to categorize all the various REs that are taking place for and in DASs.

This is the model we came up with!

What the Model is Not

There is no approach of any kind lurking in this model…

neither for specification, design, or implementation.

Purpose of the Model

The purpose of the model is to allow identifying all the various REs taking place for and in a DAS and …

to recognize that some of this RE is taking place during the execution of the DAS.

Adapt-Ready Systems

Let S_{AR} be a DAS operating on domain (set of possible inputs) D.

A *target* program S_i of S_{AR} is a program exhibiting one of the behaviors that S_{AR} can adopt after adapting.

S_i's domain is D_i.

The set of all target programs supported by S_{AR} is S.

The initial target program of S_{AR} is called S_0.

Each index i should be regarded as a name for some target program.

The only semantics that can be derived from the numerical order of the indices is the time history of target programs.

That is, there is no particular semantic relationship between S_i and S_{i+1}, other than the order of their occurrences.

Four Levels of RE

We argue that the 4 levels of RE for and in S_{AR} are:

1. **RE, done by humans, for all the target programs in S, to determine D_i for each $S_i \in S$ and S_i's reaction to each input in D_i *and* system invariants (Traditional RE),**
2. **RE, done by S_{AR} during its own execution in order to determine from the latest input that it must adapt and to determine which $S_i \in S$ to adopt (Dynamic RE),**

Four Levels of RE, Cont'd

3. **RE, done by humans, to determine S_{AR}'s adaptation *elements*, which allow S_{AR} to do the adaptation that is embodied in the Level 2 RE (RE for Adaptation Mechanisms for Specific System), and**

4. **RE, done by humans, to discover and develop adaptation mechanisms in general (General Adaptation REsearch).**

Four Levels of RE, Cont’d

The adaptation elements in Level 3 RE include

1. **monitoring techniques,**
2. **decision-making procedures, and**
3. **adaptive mechanisms.**

Four Levels of RE, Cont’d

These levels are ordered in increasing *metaness*.

Level $j+1$ RE makes decisions about the subject matter of Level j RE.

Level indices do not indicate order of occurrence.

Of course, other decompositions into levels are possible.

Concurrency of RE Levels

For a given S_{AR}, it is possible that the human RE Levels 1, 3, and 4 be done concurrently.

I.e., the human requirement engineers for S_{AR} will need to determine

- **the set of target programs,**
- **the method for choosing among them, and**
- **general monitoring and adaptation techniques**

concurrently to get a coherent system.

Redoing of RE Levels

Also, these human RE levels may need to be revisited during S_{AR}'s life.

S_{AR} may be presented totally unanticipated input $I \notin D$, such that …

S_{AR}'s Level 2 RE fails to adapt.

On Failure to Adapt

Perhaps,

- **S_{AR} informs the user that S_{AR} cannot adapt to the input I,**
- **the user must somehow notice that S_{AR} is not meeting its requirements,**
- **etc.**

Failed Adaptation, Cont'd

In such a case,

- **Level 1 RE must be done to determine at least one new target program, S_I, whose domain has *I* and that responds correctly to *I*.**
- **Level 3 RE must be done to revise S_{AR}'s adaptation mechanism so that when S_{AR} is run again with input *I*, S_{AR} does the correct, new Level 2 RE in order to adapt to *I*.**

Failed Adaptation, Cont’d

Perhaps, some Level 4 RE should be done to determine better ways to deal with unanticipated input.

One Example of a DAS

Steve Fickas *et al* (ICSE Invited Lecture) have developed an adaptive, assistive e-mail system to help brain-injured patients improve their social connectedness.

In the history of this development, we can see examples of all 4 levels of RE.

Example Level 1

Fickas *et al* did Level 1 RE to determine all possible e-mail features and UIs to be supported by any version of the e-mail system for a cognitively disabled person.

Example Level 3

Fickas *et al* did Level 3 RE to determine

- **the categories of users to be helped by the system,**
- **how to recognize a user’s category by his or her input, and**
- **the appropriate collection of features for each category of user.**

Example Level 3, Cont’d

This RE was done by a combination of

- **interviews of patients and**
- **analysis by**
  - **caretaking experts and**
  - **computing experts.**

Example Level 3, Cont'd

Patient goals were matched to …

skills need to achieve them

and then to …

features requiring those skills.

Example Level 3, Cont’d

Doing this Level 3 RE led to

- **the discovery of the need for e-mail features not anticipated in the previous Level 1 RE effort and**
- **the invention of these additional e-mail features, i.e., some more Level 1 RE.**

Example Level 2

The *e-mail system* does run-time Level 2 RE, as it

- **monitors a user’s input and**
- **determines that it is now time to change the e-mail system’s behavior to appear to the user as a new e-mail program.**

However, …

Example Level 2, Cont'd

if **the e-mail system cannot adapt to a user**

or

Fickas *et al* determine that
the user's e-mailing is deteriorating
or
the user is behaving in unanticipated ways not detected by the run-time monitoring,

then **Fickas *et al* intervene and do more Level 1 and Level 3 RE.**

Example Level 4

Fickas *et al* do Level 4 RE in the form of their research in requirements satisfaction monitoring and adaptation, requirements deferment, personal and contextual RE, etc..

Timing of Levels 3 & 2 RE

In this example and in general, …

Level 3 RE will happen before Level 2 RE simply because it is Level 3 RE that determines the Level 2 RE that S_{AR} does during its execution.

Boundary Twixt Levels 3 & 2 RE

While in any given S_{AR} the boundaries between Levels 1, 2, and 3 RE are precise, …

in a history of versions of S_{AR}, as the human requirements engineers understand better the adaptations that need to be made, …

work may shift from Levels 1 and 3 RE, done by humans, to Level 2 RE, done by the next version of S_{AR}.

Distribution of RE over Levels

In each DAS, the distribution over the 4 levels is different, and …

in some cases, some levels may even be missing.

Certainly, in a more routine case, Level 4 may be missing.

Distribution of RE, Cont’d

In a given case, Level 2 RE may be little more than conditional statements or assertions testing domain assumptions, and then deferring to human intervention for adaptation.

Another Example of a DAS

An example DAS with minimal Level 2 RE is Martin Feather’s degenerate case of an adaptive tool that he has written for himself as the only user.

He put into the tool *assert* statements, each of which causes a run-time break when its logical expression evaluates to false.

Degenerate Example, Cont’d

Each assert statement is effectively a requirements spec describing an assumed property of the input or computed value of the tool.

Often, violation of an assumption points to a requirements change; Feather is using the tool in an unanticipated way, to which the existing code is not prepared to respond in a reasonable way.

Degenerate Example, Cont'd

Feather reacts by manually modifying the code and the violated assert statements to reflect the new requirements and assumptions.

In this case, nearly all of all four levels of RE are done by Feather, the user–implementer himself.

Degenerate Example, Cont'd

Only exception:

- **the part of Level 2 RE that detects that the current input is not in the tool's current domain and that the tool's behavior must be changed.**

The rest of Level 2 RE is done off line by Feather.

Degenerate Example, Cont'd

Thus, Level 3 RE is rather trivial: need to figure out *only* logical expressions of the assert statements that monitor requirements changes.

Degenerate DAS = Robust SW

In a sense any fully robust program that checks *all* input and shuts down for any input not in its domain is a degenerate DAS like Martin Feather's.

The only way for such a program to adapt is for its authors to change it.

Extreme Example

Consider the ultimate non-human example of a fully adaptable CBS!

Extreme Example

Consider the ultimate non-human example of a fully adaptable CBS!

Commander Data, of Star Trek: Next Generation, 24th Century

But… he is a fiction!

Commander Data, of Star Trek: Next Generation, 24th Century

Although Data is a fictional character, he was conceived and written to life by technically savvy writers who managed to infuse enough consistency in his behaviors and abilities that it is possible to see how his behaviors and abilities could be programmed, given sufficiently powerful computers.

Fiction, Cont’d

Of course, current technological limitations preclude Data’s existence in any but the far distant future!

If **Moore’s law continues to hold for the next 250 years, Data might just be possible!**

Data’s Level 1 RE

is that done by Noonian Soong, Data’s inventor and builder, for the general behavior of all of his androids, including Data

Data’s Level 2 RE

is that done by Data when he recognizes a situation not covered by his current programming and past learning:

Data’s Level 2 RE, Cont’d

He simulates at positronic computer’s speed all sorts of randomly generated scenarios commencing with the current situation;

he chooses and remembers the one with the best outcome,

This simulation followed by remembering is called *adaptation and learning*.

Observation by Farhad Arbab: the look ahead must be bounded, either by time or inherently (as in chess), and the limit must be part of the Level 2 run-time RE.
Also it should be specified in the Level 3 RE, that target System i+1, produced by the Level 2 run-time RE, should be as close as possible in behavior to target System i, so the user does not experience a large, disconcerting change in behavior and user interface.

Data’s Level 3 RE

is that done by Noonian Soong to determine how Data adapts and learns.

Data’s Level 4 RE

is the research done by Noonian Soong to improve Data and other androids, e.g., to devise an emotion chip

Clarity

Whatever the distribution of levels in a particular DAS, the behavior of the DAS should be clearer to its designers if they understand the 4 levels of RE for and in it.

Details

We now describe each level of RE in detail.

Level 1 RE

Level 1 RE resembles traditional RE that is done for any CBS:

1. **eliciting and analyzing information about domain *D* of *S_{AR}*,**

2. **deciding the set of all features of any target program to be adopted by *S_{AR}* and their functionalities,**

Level 1 RE, Cont'd

3. **deciding the set of all target programs to be adopted by S_{AR} and their functionalities,**
4. **specifying the functionalities of all target programs presented by S_{AR}, and**
5. **identifying system invariants, for assurance purposes.**

A wide variety of standard methods are available for this RE.

Level 2 RE

Level 2 RE is what S_{AR} does when it gets input not in the domain of its current target program.

S_{AR} must figure out which target program in S it should adopt next.

That this behavior is RE can be seen if one considers what S_{AR} is doing.

Level 2 RE, Cont'd

Suppose S_{AR} currently has adopted the target program S_i, and its current input I is not in D_i.

Then, S_{AR} effectively

1. **determines from I how its new domain D_{i+1} differs from D_i,**
2. **determines which of its target programs, S_{i+1}, to adopt next, and**
3. **modifies its own behavior to adopt S_{i+1} as its current target program.**

Level 2 RE ⇒ Code

To do this RE, S_{AR} must have inside it

- **some code to monitor environmental changes as reflected in its input.**
- **some code that determines which of its target programs to adopt as a function of detected environmental changes.**
- **for each target program S_j, either**
  - **the code for S_j or**
  - **code to find the code for S_j, e.g., in a library.**

Level 2 RE, Cont'd

Note that all of this code has to be planned ahead of time, in what is called Level 3 RE.

Level 3 RE

Level 3 RE is probably the most difficult.

It requires assessing what S_{AR} should do at the meta level, i.e., how to make S_{AR} do its Level 2 RE.

Level 2 RE, Cont'd

Level 3 RE involves figuring out *how* to get S_{AR} to

1. **determine from *I* how its new domain D_{i+1} differs from D_i,**
2. **determine which of its target programs, S_{i+1}, to adopt next, and**
3. **modify its own behavior to to adopt S_{i+1} as its current target program.**

Level 2 RE, Cont’d

Doing this RE requires having determined program-testable correspondences to environmental changes that trigger adaptation. The requirements engineers will have to explore representations for

1. **the possible new domains with their corresponding environmental conditions,**
2. **the possible adaptive reactions to new inputs, and**
3. **the testable conditions under which each new adaptive reaction is to be applied.**

Level 2 RE, Cont’d

Representations can be any scheme from which specific reactions can be derived, perhaps by

- **instantiation,**
- **parameter application,**
- **mapping,**
- **table lookup,**
- **formula,**
- **specification generation,**
- **or cetera**

RE Questions for Level 3 RE

The RE questions to be addressed by S_{AR}, during its execution, are:

- **What sort of unexpected input warrants adaptation?**
- **What adaptive action is appropriate in response to the unexpected input?**
- **What kind of decision-making system should be used to determine the adaptation? (rule-based, AI-based, etc.)**

Level 4 RE

Level 4 RE is essentially the research into adaptation mechanisms.

Adaptation mechanisms have been developed for

- **the application level,**
- **middleware, and**
- **operating systems.**

Limit of Adaptability

The Command^{er} Data Example drives home an important point:

A DAS *S* can be no more adaptable than our own ability to identify the adaptations that *S* might need to make.

For the foreseeable future, software is not able to think and be truly intelligent and creative.

Limit, Cont'd

Therefore, the extent to which *S* is able to adapt is limited by ...

the extent to which *S*'s human programmers planned for *S*'s adaptability.

This limit is called the *envelope of adaptability*.

Limit, Cont'd

***S'*s envelope of adaptability cannot exceed our own adaptability.**

While we are adaptable, ...

we do not know how or why we are adaptable.

Thus, we cannot program software to be even *as* adaptable as we are.

Therefore, *S* will always be less adaptable than we are.

Belief Suspension

Some talk about DASs adapting to design faults!

Clearly, a DAS, ultimately programmed by humans, is not able to diagnose and fix true design faults.

For a DAS to fix any fault, the humans implementing the DAS have to have anticipated the fault, and …

if a fault is anticipated, it is not a design fault.

Reality

The phone company’s switching system has true design faults that occasionally rear their ugly heads.

The system’s response to a design fault is

- **to report the fault and all diagnostic data it can find and *then***
- **to quickly mask the fault, for minimal disruption of the system.**

Phone Switching System, Cont’d

Then humans investigate the fault offline, using the diagnostic information.

When the humans have located the source of the fault, they

- **fix the code,**
- **test it offline, and then**
- **install the fix with as little disruption as possible.**

Phone Switching System, Cont'd

In other words, a DAS does not truly detect and fix a design fault.

The DAS only masks the fault and …

lets humans determine the source of the fault and fix it.

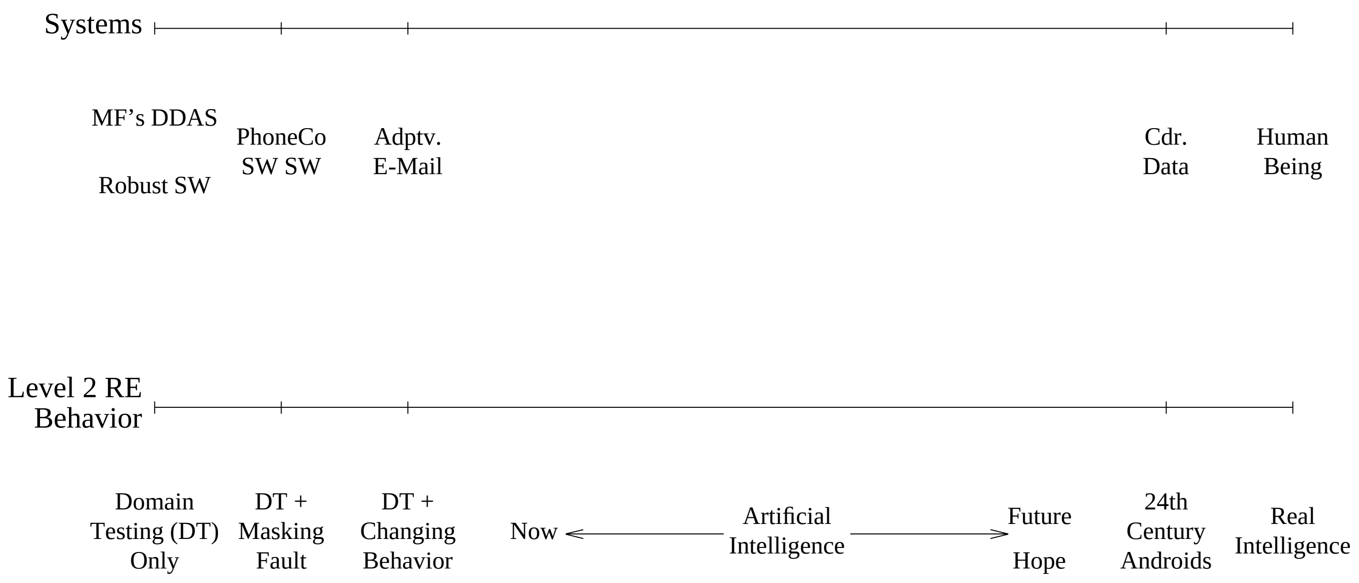
Spectrum of DASs

We can now see a spectrum of DASs.

The variation is over how much Level 2 RE is done by the DAS at run time, i.e. the amount of *dynamism*.

It can vary from just domain testing to complete intelligence, with, masking, adaptation, and artificial intelligence in the middle.

Logarithmic Scale of Dynamism



Minimally Adaptive DAS

I would not begin to call a DAS “adaptive” unless it is doing at least some behavior change.

That is, in my view, masking faults is not really adapting.

What’s Next?

Costs for CBSs are decreasing.

Demand for mobile, heterogeneous, and pervasive systems is increasing.

Interest in autonomic systems is increasing.

Therefore, the need for DASs will increase.

Next, Cont’d

As more and more systems become adaptive, …

we believe that the adaptability envelope will expand …

since the RE at Level 1 will expand to include RE at Levels 3 and 4.

Next, Cont’d

Therefore, more attention will be needed to establish the correctness of software,

- **before,**
- **during, and**
- **after**

adaptation.

Next, Cont’d

Thus far the focus has been on *enabling* adaptation.

We need to consider assurance issues at all 4 levels of RE for DASs.

Assurance will contribute also to the determination of when, how, and where adaptations should take place.

This is the focus of author Zhang’s research.

Conclusions

We presented this talk at the DEAS workshop, and found that the model fits *every* DAS described at the workshop.

Based on the comments there and the comments here, we plan to write a journal paper describing the model more completely, and applying it to describe many DASs.