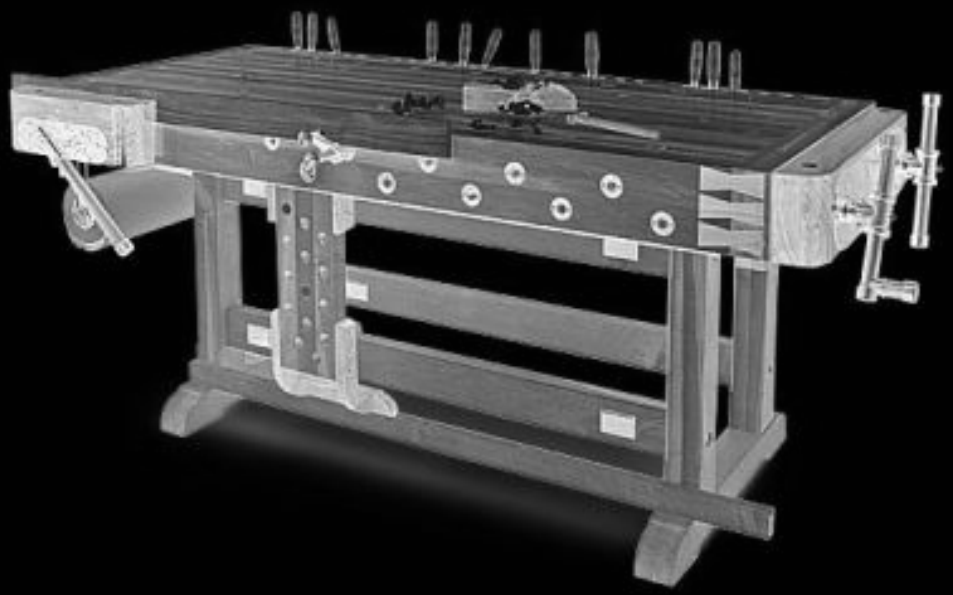
Models, Modeling and Tools in Embedded Software Development



voelter { ingenieurbüro für softwaretechnologie // itemis

voelter@acm.org
www.voelter.de
@markusvoelter

1 Motivation

2 What is a Model?

3 Characteristics of Models

4 Example Models Languages

5 Performance and Resources

6 Reliability

7 Language Workbenches

8 JetBrains MPS

9 mbeddr

10 Recap/Big Picture

Philosophical

Practical

Phil

1 Motivation

2 What is a Model?

3 Characteristics of Models

4 Example Models Languages

5 Performance and Resources

6 Reliability

7 Language Workbenches

8 JetBrains MPS

9 mbeddr

10 Recap/Big Picture

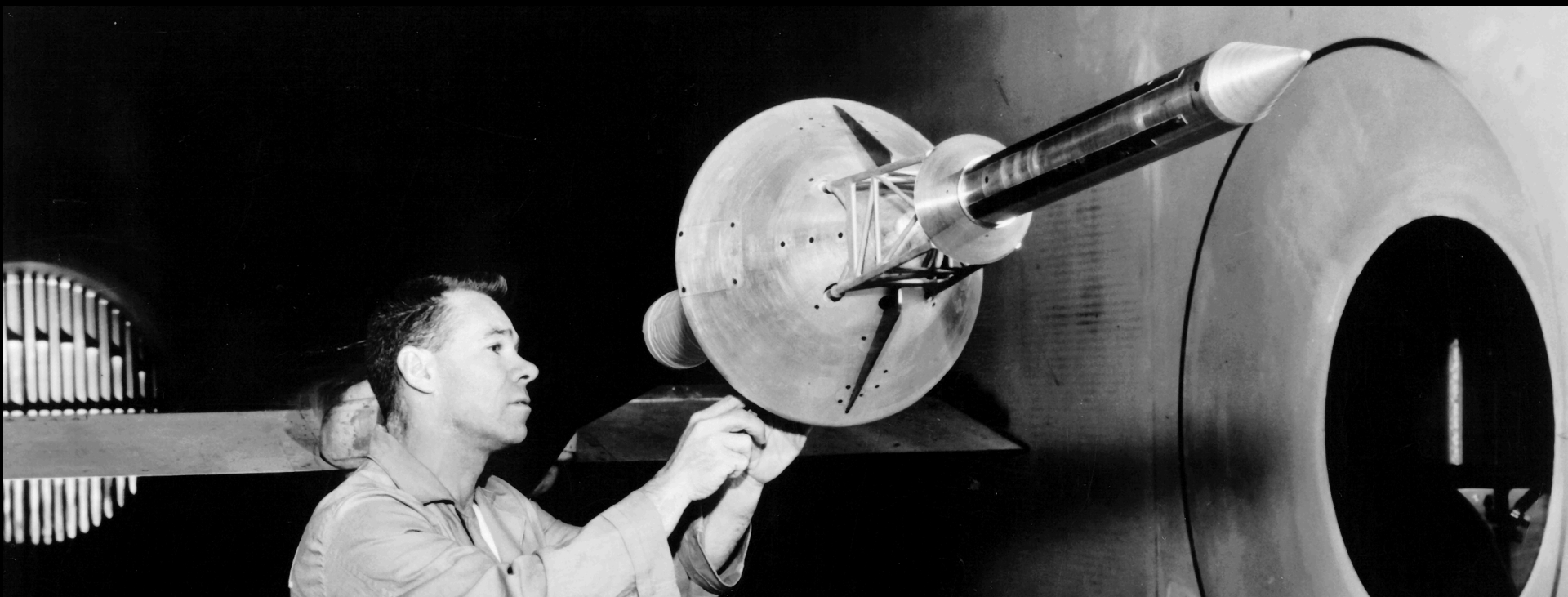
1

Motivation

Complexity is killing us.

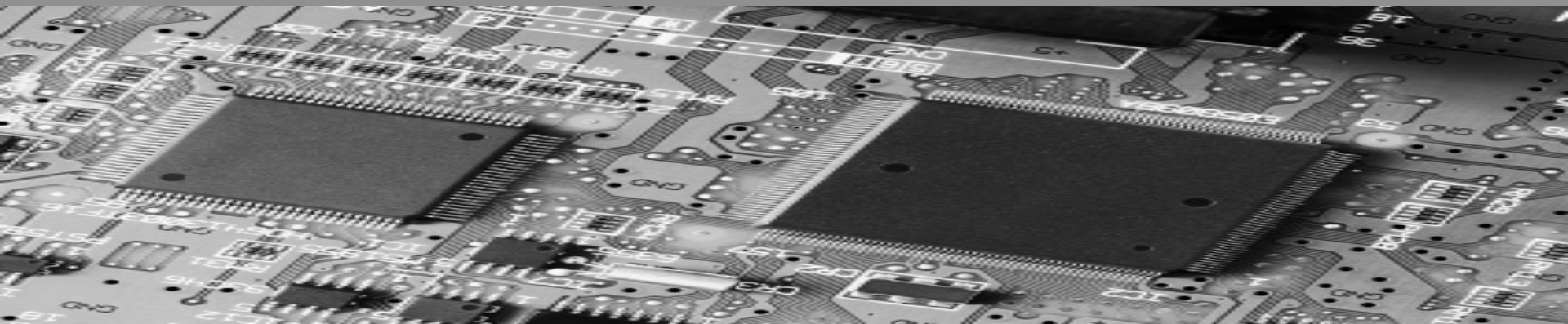
So?

(Software) Engineering to the Rescue!



Modularization

Procedures, Classes, Components, Services, User Stories



Encapsulatation

Private Members
Frameworks
Facade Pattern
Components
Layers/Rings/Levels
Packed Data Wrapper

Contracts

Interfaces
Pre/Post Conditions
Protocol State Machines
Message Exchange Patterns
Published APIs

Cross-Cutting Concerns

Aspect Orientation
Interceptors
Application Servers
Exception Handling

Parametrization

Function Arguments
Command-Line Args
Configuration Files

Types & Instances

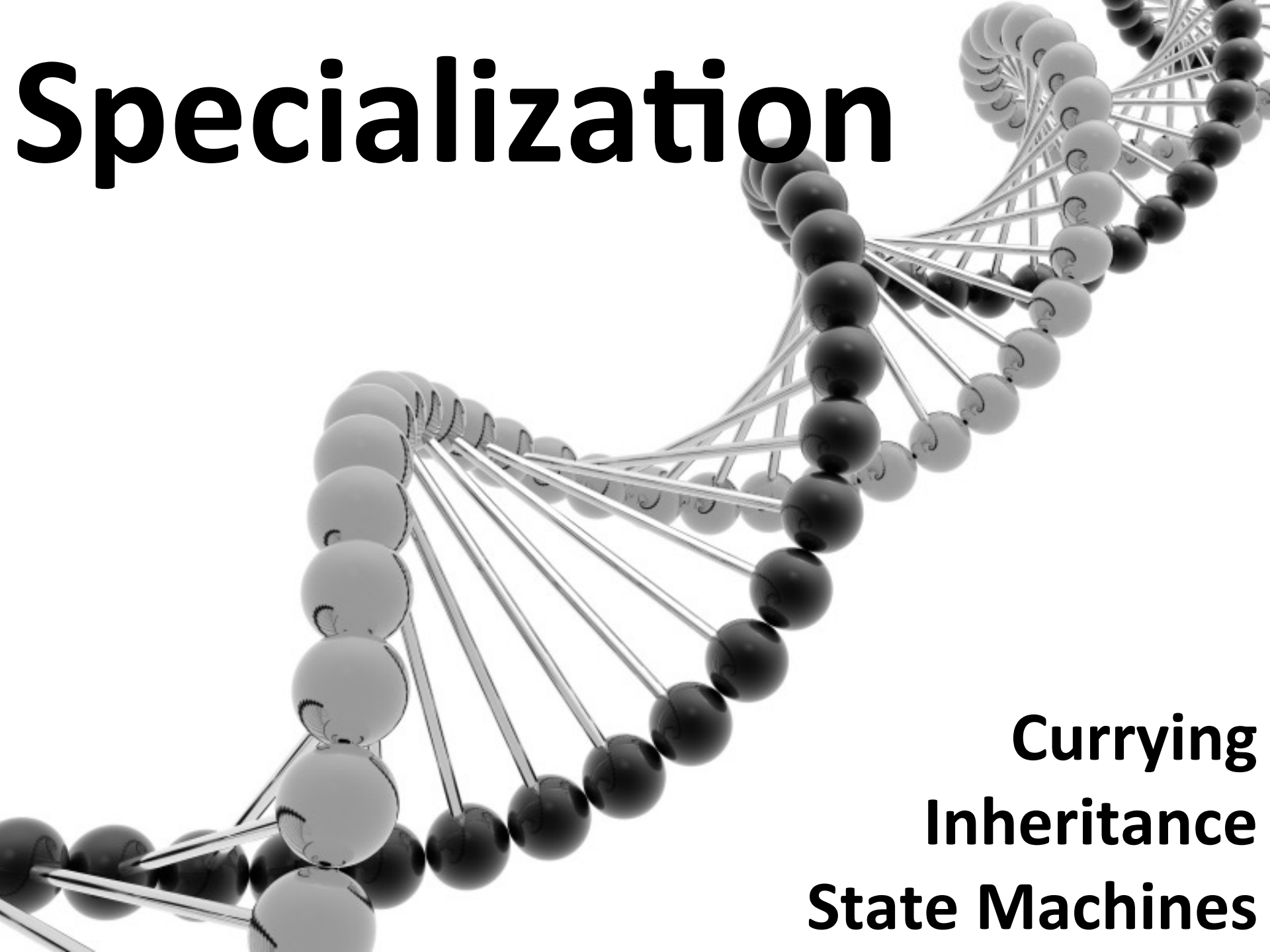
Programming Languages
Components

Models & Metamodels
RDBMS/XML Schemas

Hierarchical Decomposition

Procedures/Methods
State Machines, Components

Specialization



Currying
Inheritance
State Machines

Viewpoints

Configuration Files
4+1 Model
Blackbox/Whitebox
Types/Instances/Deployment

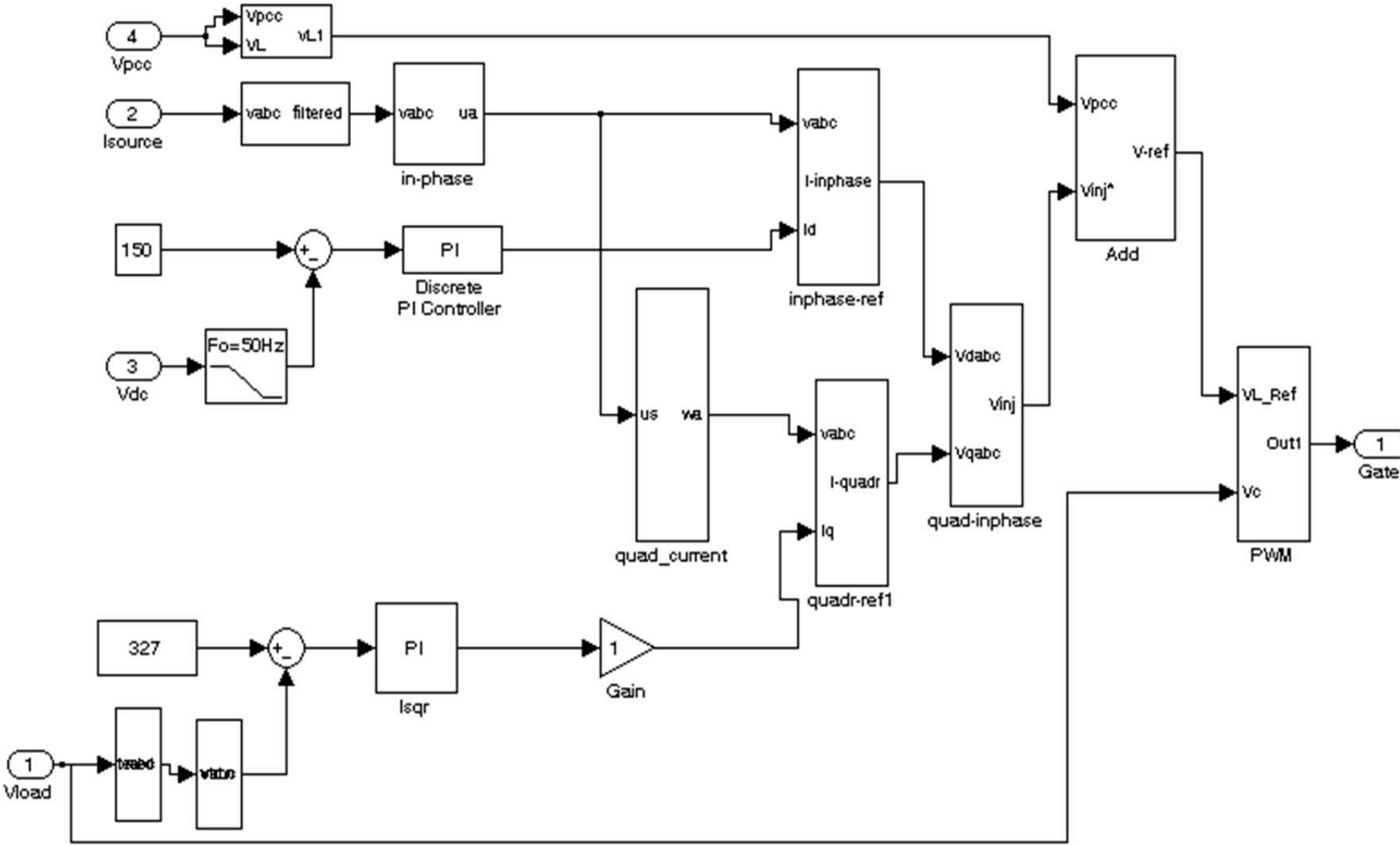
Declaration

Implementation

App Servers (EJB), Plugin RT (Eclipse)
Models, Transactional Memory

<Reality Check>

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;       /* Encode function number 0->none */
  ULONG  key;           /* Key for function (N/A if no  func) - max key
                        value=64k-1  */
  ULONG  length;        /* Length of following data */
  ULONG  start          /* start position of the encrypted data */
  ULONG  cntrlblock[20] /* control block */
  UCHAR  data[500];     /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k) (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k) (UCHAR)((UCHAR)(~d) ^ (UCHAR)k)
#define ENCODE3(d,k) (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k) (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k) (UCHAR)(~(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k) (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;     /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:        NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                                /* Function number */
```


4
Vpcc
Vpcc
VL
vL1
2
Isource
vabc filtered
vabc ua
in-phase
150
PI
Discrete
PI Controller
3
Vdc
Fo=50Hz
vabc
I-inphase
Id
inphase-ref
us wa
quad_current
vabc
I-quadr
Iq
quadr-ref1
Vdabc
Vinj
Vqabc
quad-inphase
Vpcc
V-ref
Vinj^
Add
VL_Ref
Out1
Vc
PWM
1
Gate
327
PI
Isqr
1
Gain
1
Vload

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

```
/*  Record structure */
typedef struct btrec    /* record */
{
  ULONG  funcnum;        /* Encode function number 0-none */
  ULONG  key;            /* Key for function (N/A if no  func) - max key
                         value=64K-1  */
  ULONG  length;         /* Length of following data */
  ULONG  start           /* start position of the encrypted data */
  ULONG  ctrlblock[20]  /* control block */
  UCHAR  data[500];      /* Data */
} BTREC;
typedef BTREC FAR * PBTREC;

/*  Encryption and decryption functions
#define ENCODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define ENCODE2(d,k)  (UCHAR)((UCHAR)(-d) ^ (UCHAR)k)
#define ENCODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)
#define DECODE1(d,k)  (UCHAR)(d ^ (UCHAR)k)
#define DECODE2(d,k)  (UCHAR)(-(UCHAR)(d ^ (UCHAR)k))
#define DECODE3(d,k)  (UCHAR)(d ^ (UCHAR)~k)

/* Name:NewKey */
/* Description: Returns a key to use in the translation */

USHORT NewKey(VOID)
{

  USHORT key;      /* Key */


/* Work out the key from a random number */
  key=(USHORT)rand();
/* return random number */
  return(key);
}

/* Name:         NewFunc */
/* Description: Returns a function number to use in the translation */
ULONG NewFunc(VOID)
{
  ULONG func;                               /* Function number */
```

</Reality Check>

THE QUESTION

How can we deal more effectively with...

Modules
Contracts
Types
Hier. Decomposition
Specialization
Viewpoints
Declarativeness
Cross-cutting Concerns

How can we also deal effectively with...

Safety
Security
Performance
Timing
Reliability
Maintainability
...
<ComeUpWithYourOwn>-ility

THE ANSWER

Your own?

Suitable Abstractions

Your own?

Suitable Notations

(Good Tools)

Models

Your own? Modeling Languages

Your own? Modeling Tools

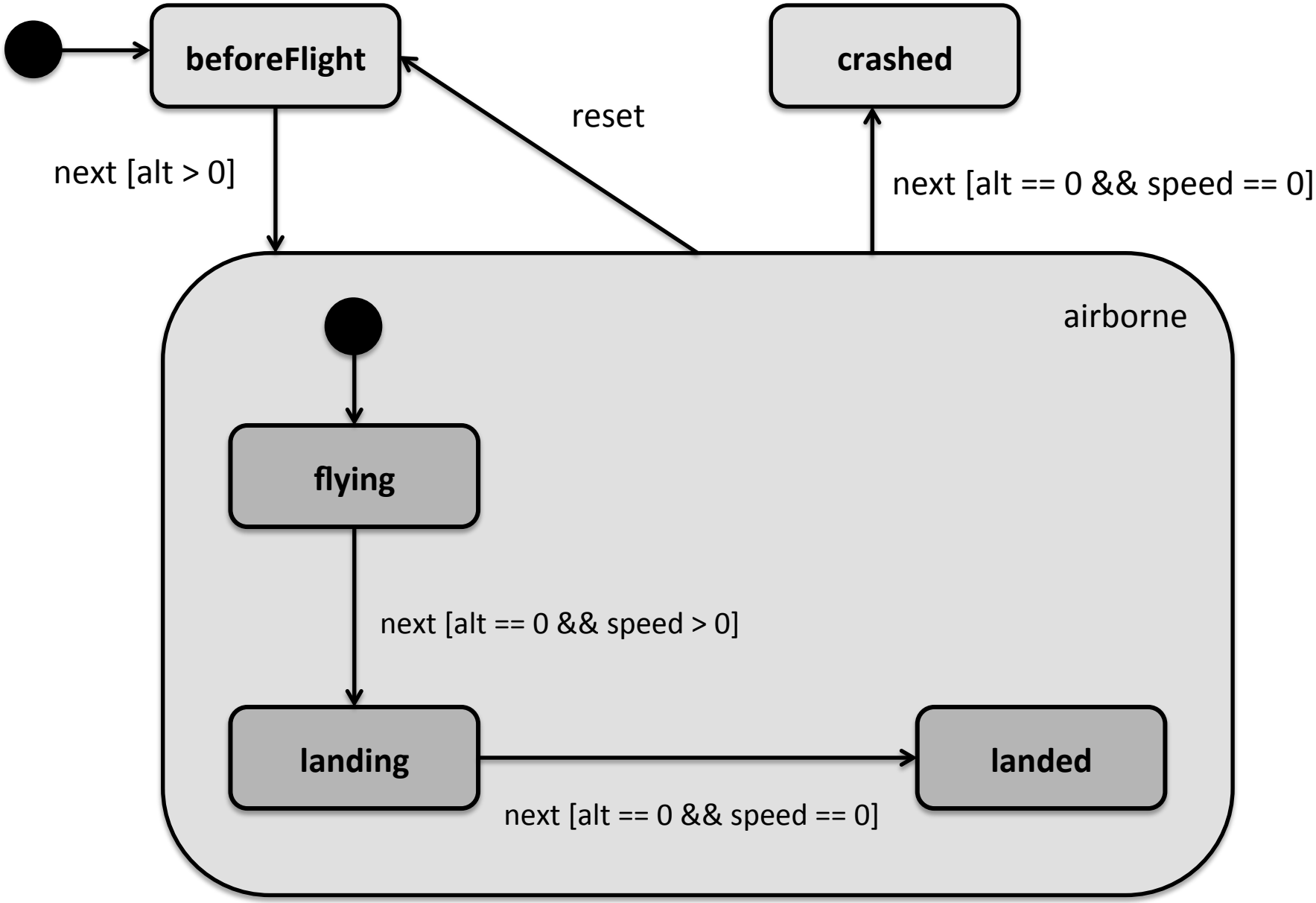
2

What is a Model?

[Model or Code]

```
Trackpoint* makeTP(uint16 alt, int16 speed) {
  static int8 trackpointCounter = 0;
  trackpointCounter++;
  Trackpoint* tp = ((Trackpoint*) malloc(sizeof Trackpoint));
  tp->id = trackpointCounter;
  tp->timestamp = trackpointCounter;
  tp->alt = alt
  tp->speed = speed
  return tp;
}
```

[Model or Code]



[Model or Code]

```
statemachine HierarchicalFlightAnalyzer initial = beforeFlight {
  in next()
  in reset()
  out crashNotification() -> raiseAlarm
  state beforeFlight {
    on next [tp->alt > 0 m] -> airborne
  }
  composite state airborne initial = flying {
    on reset [ ] -> beforeFlight
    on next [tp->alt == 0 m && tp->speed == 0 mps] -> crashed
    state flying {
      on next [tp->alt == 0 m && tp->speed > 0 mps] -> landing
      on next [tp->speed > 200 mps] -> airborne
      on next [tp->speed > 100 mps] -> airborne
    }
    state landing {
      on next [tp->speed == 0 mps] -> landed
      on next [ ] -> landing
    }
    state landed {
    }
  }
  state crashed {
  }
}
```

[Model or Code]

```
statemachine HierarchicalFlightAnalyzer initial = beforeFlight {
  in next(Trackpoint* tp)
  in reset()
  out crashNotification() -> raiseAlarm
  readable var int16 points = 0
  state beforeFlight {
    on next [tp->alt > 0 m] -> airborne
    exit { points += TAKEOFF; }
  }
  composite state airborne initial = flying {
    on reset [ ] -> beforeFlight { points = 0; }
    on next [tp->alt == 0 m && tp->speed == 0 mps] -> crashed
    state flying {
      on next [tp->alt == 0 m && tp->speed > 0 mps] -> landing
      on next [tp->speed > 200 mps] -> airborne { points += VERY_HIGH_SPEED; }
      on next [tp->speed > 100 mps] -> airborne { points += HIGH_SPEED; }
    }
    state landing {
      on next [tp->speed == 0 mps] -> landed
      on next [ ] -> landing { points--; }
    }
    state landed {
      entry { points += LANDING; }
    }
  }
  state crashed {
    entry { send crashNotification(); }
  }
}
```

Does it really matter?

What is the difference?

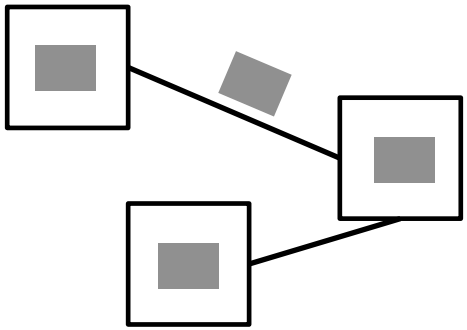
Who cares?

[Notation]

One important aspect of languages.

Textual

Graphical

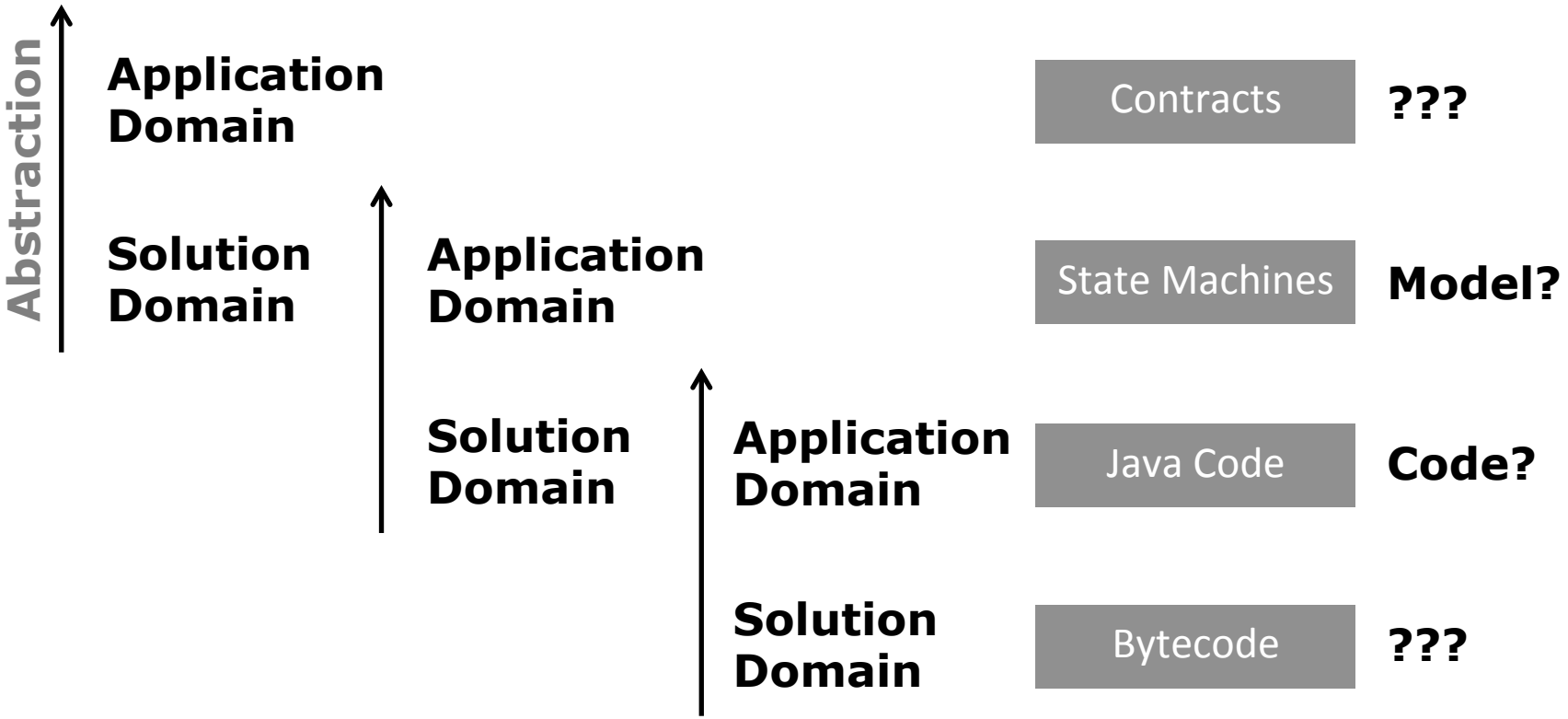


Tabular

Mathematical

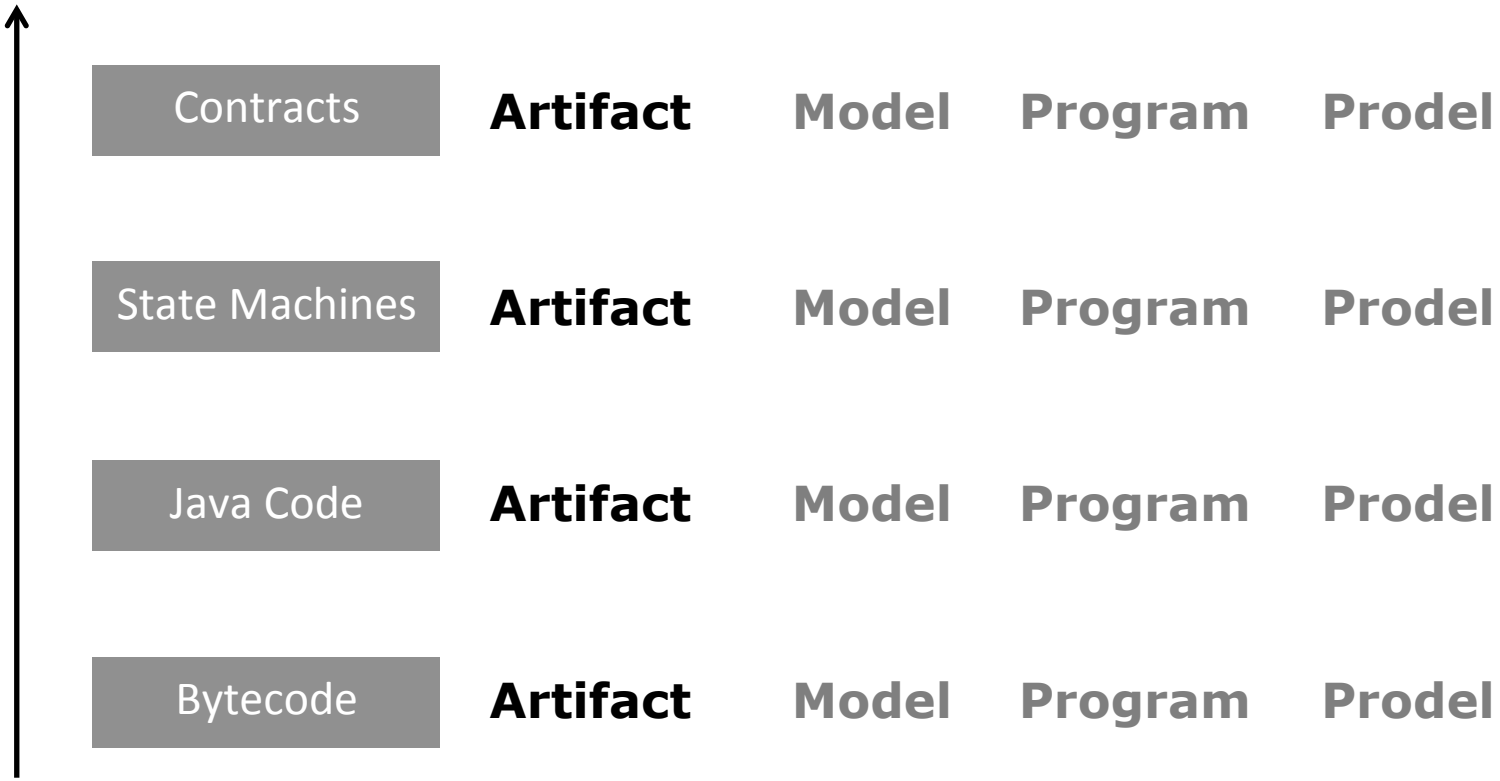
[Abstraction]

The other important aspect.

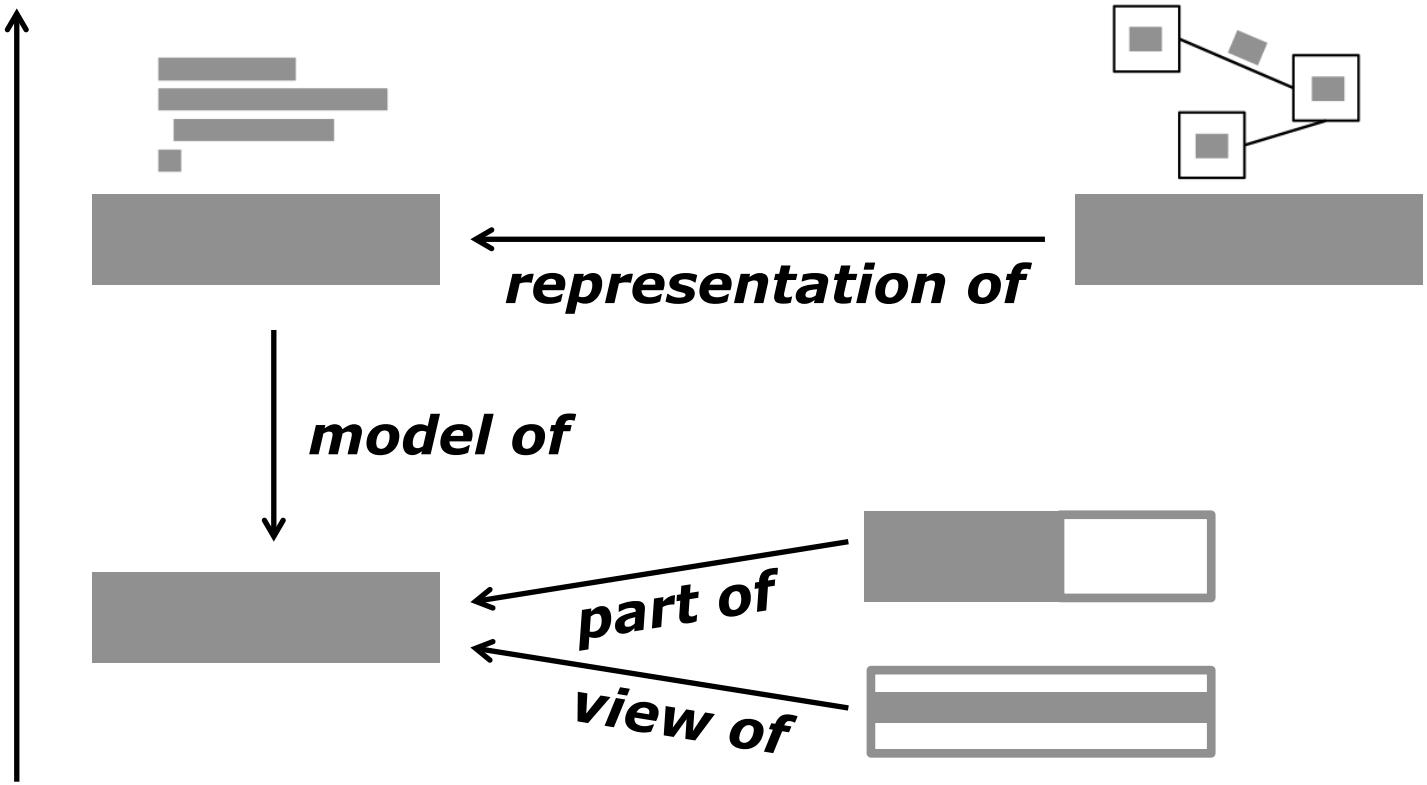


[Abstraction]

The other important aspect.



[Relationships]



[Relationships]

model of Abstraction

part of Hierarchy, Partitioning

view of Concern, Aspect, Viewpoint

representation of
Presentation, Notation, Form

3

Characteristics of Models

Dimensions of Models

Abstractions

Notations

Purpose

Standardization

Tool Support

Aligned with Domain

Driven By Model Purpose

SE Best Practices

Meaningful to Users

Well-defined Semantics

One Size does NOT fit all.

-> Multi-Paradigm!

Dimensions of Models

Abstractions

Notations

Purpose

Standardization

Tool Support

Text, Tables, Math, ...

Inspired by Existing N

Scalable!

Integrate Multiple

Switchable for same Abstr.

More than Diagrams

Dimensions of Models

Abstractions

Notations

Purpose

Standardization

Tool Support

Create-and-Think

Communicate

Stakeholder-Programming

Analysis

Interpretation/Generation

- Effort
- Platform
- Complexity

Dimensions of Models

Abstractions

Notations

Purpose

Standardization

Tool Support

Avoid Reinventing Wheels

Enables Data Exchange

Makes Hiring Simpler

Limited Domain Match

Potentially Large

Standardize on Meta Level

Adopt a Language

Dimensions of Models

Abstractions

Notations

Purpose

Standardization

Tool Support

Absolutely Crucial!

As good as IDE!

Standards help, but ...

Language Workbenches

Integrate L, not T

4

Example Models Languages

Requirements

A mix of text and structure
With real cross-references

```
FlightJudgementRules
doc config: Config      filters:
class:      default     imports: ArchitecturalComponents
These are the requirements for judging flights.

1 Initially you have no points.
  InitialNoPoints /functional: tags
  When the game starts, you have no points.
  workpackage inital  scope: 1 responsible: peter prio: 1 effort: 1 days

2 Once a flight lifts off, you get 100 points
  PointsForTakeoff /functional: tags
  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent feugiat enim arcu, ut egestas velit. Suspendisse
  potenti. Etiam risus ante, bibendum ut mattis eget, convallis sit amet nunc. Ut nec justo sapien, vel condimentum
  velit. Quisque venenatis faucibus tellus consequat rhoncus. Vestibulum dapibus dictum vulputate. Phasellus rhoncus
  quam eu dui dictum sollicitudin.
  Duis tempus justo magna. Nunc lobortis libero sed eros interdum aliquet eleifend nisl mattis. Vivamus est orci, tempus
  non sagittis sed, vulputate quis nunc. Integer sollicitudin enim in orci iaculis facilisis non et elit. Fusce rutrum,
  eros faucibus congue scelerisque, sapien sapien pharetra leo, quis rhoncus velit enim vel orci. Etiam id elit leo,
  ultricies viverra mauris. Pellentesque pretium dui varius eros sodales tempus. Sed nec arcu eu lectus euismod sodales
  sit amet consectetur augue. Aliquam nibh arcu, egestas sit amet interdum quis, sollicitudin et eros. Pellentesque non
  lectus a lacus sollicitudin pellentesque et sed metus. Integer metus urna, semper sit amet sollicitudin vel, ultrices
  vel massa. Suspendisse id auctor turpis. Curabitur interdum blandit nisi, a varius ipsum tempor et.
  Fusce vitae quam est, at facilisis
  requires also ExampleWithDependencies
  constant int8 POINTSFORTAKEOFF = 100
  workpackage impl1  scope: 1 responsible: peter prio: 1 effort: 10 days
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent feugiat enim arcu, ut egestas velit. Suspendisse
    potenti. Etiam risus ante, bibendum ut mattis eget, convallis sit amet nunc. Ut nec justo sapien, vel condimentum
    velit. Quisque venenatis faucibus tellus consequat rhoncus. Vestibulum dapibus dictum vulputate. Phasellus rhoncus
    quam eu dui dictum sollicitudin. vestibulum mollis, magna sapien eleifend odio, vel sagittis mi dolor vitae metus.
    Fusce at v
  workpackage impl2  scope: 2 responsible: peter prio: 1 effort: 5 days
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent feugiat enim arcu, ut egestas velit. Suspendisse
    potenti. Etiam risus ante, bibendum ut mattis eget, convallis sit amet nunc. Ut nec justo sapien, vel condimentum
    velit. Quisque venenatis faucibus tellus consequat rhoncus. Vestibulum dapibus dictum vulputate. Phasellus rhoncus
    quam eu dui dictum sollicitudin. Now I can edit it here directly.
    @node[ExternalFunctions]
```

Requirements

Formal/Structured specs can be embedded and evaluated

1.2 Flight is Interpolated and Evaluated

FlightIsInterpolated /use case: tags

Lorem ipsum dolor sit amet, consectetur adipiscing elit. M
potenti. Etiam risus ante, bibendum ut mattis eget, convallis sit amet nunc. Ut nec justo sapien, vel condimentum velit. Quisque venenatis faucibus tellus consequat rhoncus. Vestibulum dapibus dictum vulputate. Phasellus rhoncus quam eu dui dictum sollicitudin.

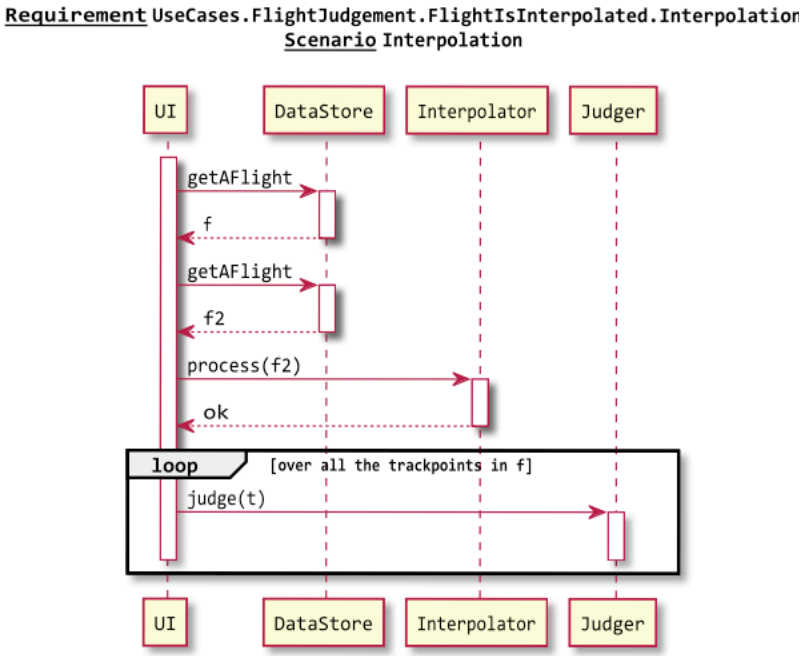
```
use case FlightIsInterpolated {
  active participant UI
  passive participant DataStore
  passive participant Interpolator
  passive participant Judger
}
```

1.2.1 Describes the Interpolation

Interpolation /scenario: tags

Text

```
scenario Interpolation
  UI {
    -> DataStore.getAFlight(): new Flight f
    -> DataStore.getAFlight() {
      return new Flight f2
    } DataStore.getAFlight
    -> Interpolator.process(received f2): ok
    loop over all the trackpoints in f {
      -> Judger.judge(new Trackpoint t)
    } loop
  }
```



Tracing

Implementation Models or Code can be trace to requirements to aid validation.

```
#constant TAKEOFF = 100; -> implements PointsForTakeoff
#constant HIGH_SPEED = 10; -> implements FasterThan100
#constant VERY_HIGH_SPEED = 20; -> implements FasterThan200
#constant LANDING = 100; -> implements FullStop

[checked]
exported statemachine FlightAnalyzer initial = beforeFlight {
```

		Events	
		next(Trackpoint* tp)	reset()
States	beforeFlight	[tp->alt > 0 m] -> airborne	
	airborne	[tp->alt -> implements FullStop == 0 m && tp->speed == 0 mps] -> crashed [tp->alt == 0 m && tp->speed > 0 mps] -> landing [tp->speed > 200 mps && tp->alt == 0 m] -> airborne -> implements FasterThan200 { points += VERY_HIGH_SPEED; }	[ ] -> beforeFlight
	landing	[tp->speed > 0 mps] -> landing -> implements ShortLandingRoll { points--; }	[ ] -> beforeFlight
	landed		[ ] -> beforeFlight
	crashed		
	NextRound <j>		

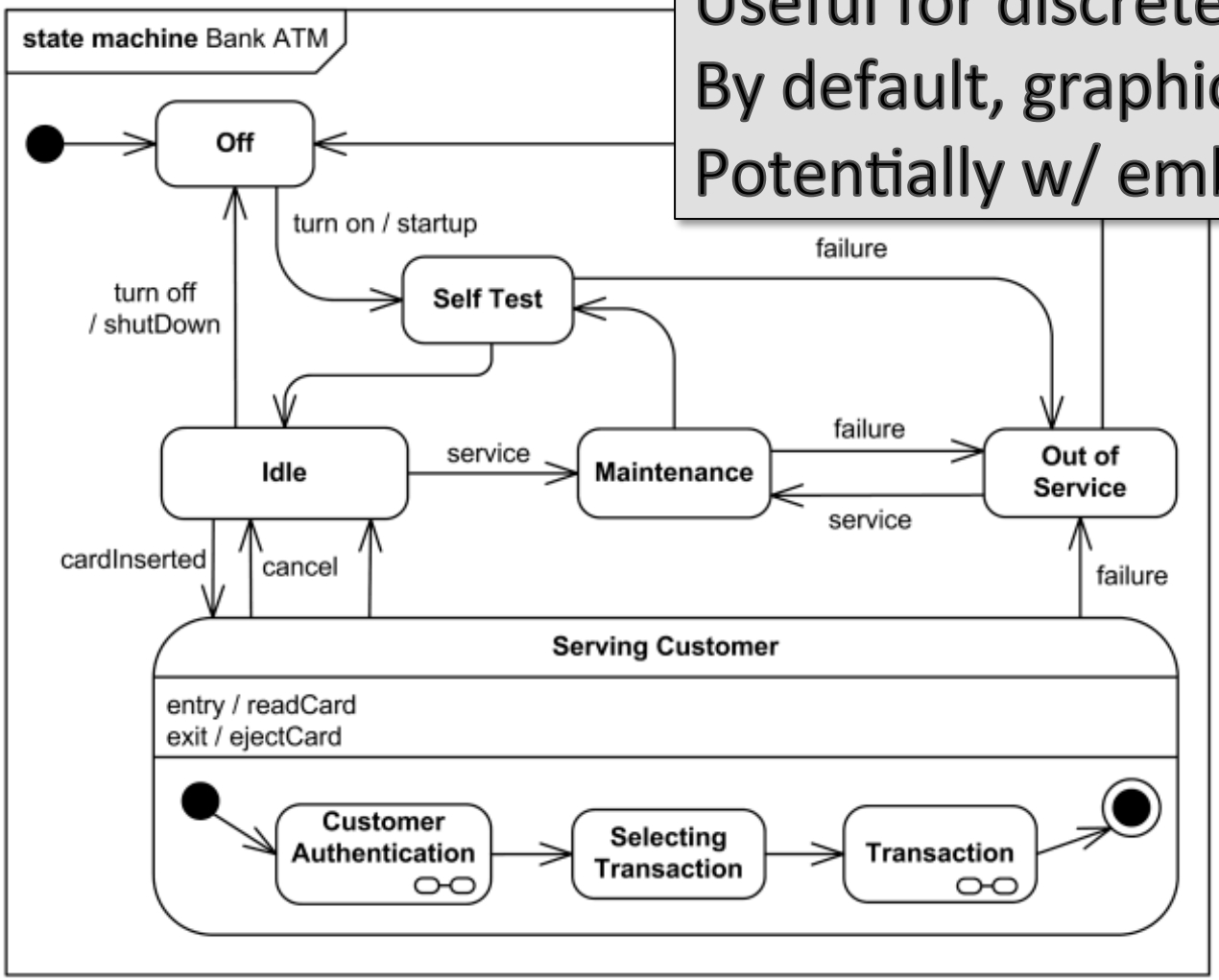
You should land as short as possible

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent feugiat enim arcu, ut egestas velit. Suspendisse potenti. Etiam risus ante, bibendum ut mattis eget, convallis sit amet nunc. Ut nec justo sapien, vel condimentum velit. Quisque venenatis faucibus tellus consequat rhoncus. Vestibulum dapibus dictum vulputate. Phasellus rhoncus quam eu dui dictum sollicitudi

```
}
```

Behavior: State Machines

Useful for discrete behavior
By default, graphical notation
Potentially w/ embedded actions



Behaviour: State Machines

State Machines may also be represented as tables.

```
#constant TAKEOFF = 100; -> implements PointsForTakeoff
#constant HIGH_SPEED = 10; -> implements FasterThan100
#constant VERY_HIGH_SPEED = 20; -> implements FasterThan200
#constant LANDING = 100; -> implements FullStop

[checked]
exported statemachine FlightAnalyzer initial = beforeFlight {
```

		Events	
		next(Trackpoint* tp)	reset()
States	beforeFlight	[tp->alt > 0 m] -> airborne	
	airborne	[tp->alt -> implements FullStop == 0 m && tp->speed == 0 mps] -> crashed [tp->alt == 0 m && tp->speed > 0 mps] -> landing [tp->speed > 200 mps && tp->alt == 0 m] -> airborne -> implements FasterThan200 { points += VERY_HIGH_SPEED; }	[ ] -> beforeFlight
	landing	[tp->speed > 0 mps] -> landing -> implements ShortLandingRoll { points--; }	[ ] -> beforeFlight
	landed		[ ] -> beforeFlight
	crashed		
	NextRound <j>		

You should land as short as possible

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent feugiat enim arcu, ut egestas velit. Suspendisse potenti. Etiam risus ante, bibendum ut mattis eget, convallis sit amet nunc. Ut nec justo sapien, vel condimentum velit. Quisque venenatis faucibus tellus consequat rhoncus. Vestibulum dapibus dictum vulputate. Phasellus rhoncus quam eu dui dictum sollicitudi

```
}
```

Behaviour: State Machines

```
statemachine HierarchicalFlightAnalyzer initial = beforeFlight {
  in next()
  in reset()
  out crashNotification() -> raiseAlarm
  state beforeFlight {
    on next [tp->alt > 0 m] -> airborne
  }
  composite state airborne initial = flying {
    on reset [ ] -> beforeFlight
    on next [tp->alt == 0 m && tp->speed == 0 mps] -> crashed
    state flying {
      on next [tp->alt == 0 m && tp->speed > 0 mps] -> landing
      on next [tp->speed > 200 mps] -> airborne
      on next [tp->speed > 100 mps] -> airborne
    }
    state landing {
      on next [tp->speed == 0 mps] -> landed
      on next [ ] -> landing
    }
    state landed {
    }
  }
  state crashed {
  }
}
```

... or as text. After all, that's just a *representation*! The model does not change!

Behaviour: State Machines

```
statemachine HierarchicalFlightAnalyzer initial = beforeFlight {
  in next(Trackpoint* tp)
  in reset()
  out crashNotification() -> raiseAlarm
  readable var int16 points = 0
  state beforeFlight {
    on next [tp->alt > 0 m] -> airborne
    exit { points += TAKEOFF; }
  }
  composite state airborne initial = flying {
    on reset [ ] -> beforeFlight { points = 0; }
    on next [tp->alt == 0 m && tp->speed == 0 mps] -> crashed
    state flying {
      on next [tp->alt == 0 m && tp->speed > 0 mps] -> landing
      on next [tp->speed > 200 mps] -> airborne { points += VERY_HIGH_SPEED; }
      on next [tp->speed > 100 mps] -> airborne { points += HIGH_SPEED; }
    }
    state landing {
      on next [tp->speed == 0 mps] -> landed
      on next [ ] -> landing { points--; }
    }
    state landed {
      entry { points += LANDING; }
    }
  }
  state crashed {
    entry { send crashNotification(); }
  }
}
```

The amount of detail may vary – depending on the purpose (analysis vs. generation)

Verification: State Machines

Temporal Logic provides a powerful formalism for verifying properties of state machines.

core.dev - [L:\lwes-assembla\mbeddr\code\languages\com.mbeddr.core] - semaphore\Ser

File Edit Search View Go To Code Build Run Tools Version Control Window Help

TypeSizeConfiguration | Semaphore

```
requirements modules: bla
module Semaphore from semaphore imports nothing {

  verifiable statemachine statemachine {
    in request(boolean req) <no binding>
       step(int[-10..10] stepCount) <no binding>
    out out(int[0..2] traffic, boolean pedestrian, boolean indicator) => handleOutE
    vars int[-1..10] counter = 0
         int[0..5] green_val = 2
         int[0..5] yellow_val = 2
         int[0..5] red_val = 4
    states (initial = Init)
      always reachable state Init {
        on step [counter == 0] -> green {
          send out(2, false, true);
          counter = 5;
        }
        on step [counter > -1 && counter < 1] -> green {
          send out(2, false, true);
          counter = 5;
        }
      }
      always reachable state green {
        on request [counter == -1] -> green {
          send out(0, false, true);
          counter = green_val;
        }
        on step [counter > 0] -> green {
          send out(0, false, true);
          counter = counter - 1;
        }
```

State 'Red' can b...	SUCCESS	
State 'RedYellow' ...	SUCCESS	
State 'Init' is not li...	FAIL	3
State 'green' is live	SUCCESS	
Variable 'counter'...	SUCCESS	
Variable 'green_v...	SUCCESS	
Variable 'yellow_v...	SUCCESS	
Variable 'red_val' ...	SUCCESS	
State 'Init' contain...	FAIL	3
State 'green' has ...	SUCCESS	
State 'Yellow' has...	SUCCESS	
State 'Red' has d...	SUCCESS	
State 'RedYellow' ...	SUCCESS	
Transition 0 of st...	SUCCESS	
Transition 1 of st...	FAIL	1
Transition 0 of st...	FAIL	1

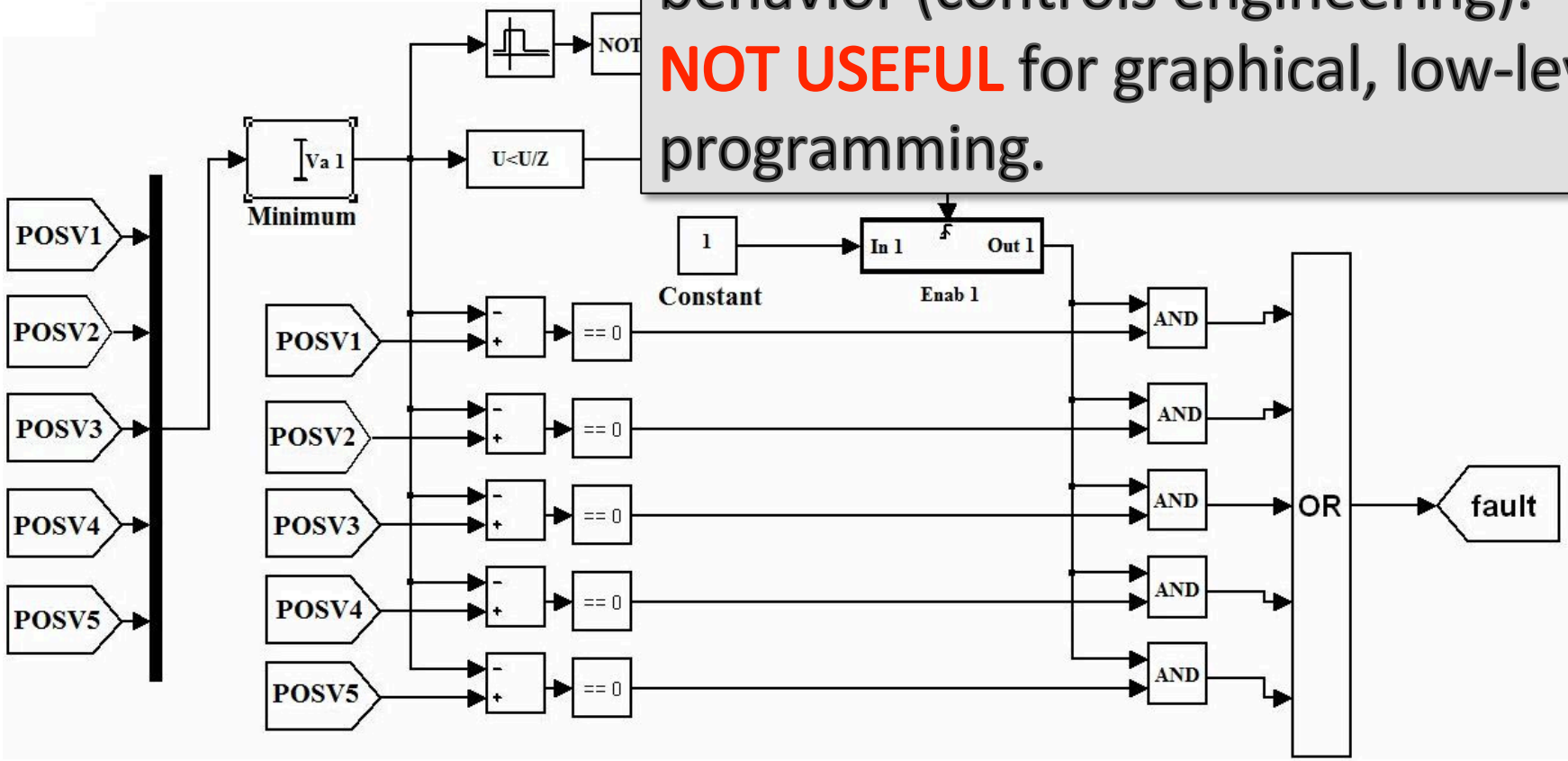
Node	Value
State Init	
counter	0
green_val	2
yellow_val	2
red_val	4
State Init	
in_event: step	step(-10)
counter	0
green_val	2
yellow_val	2
red_val	4
State green	
out_event:out	out(2, false, true)
counter	5
green_val	2
yellow_val	2
red_val	4

Version Control | 0: Messages | 9: Changes | 2: Inspector

172M of 1190M

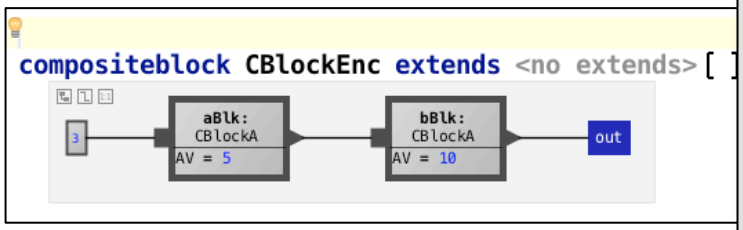
Behavior: Dataflow Models

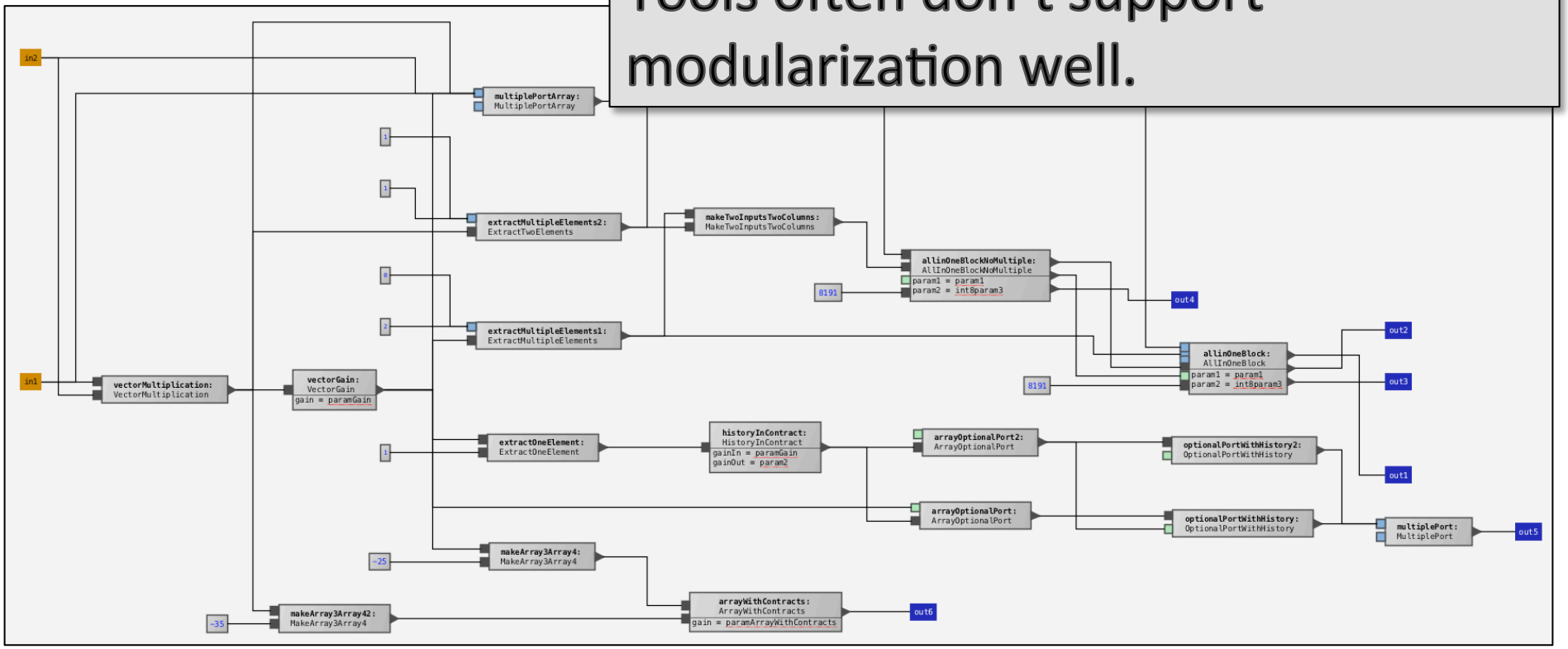
Useful for implementing continuous behavior (controls engineering).
NOT USEFUL for graphical, low-level programming.



Behavior: Dataflow Models

Dataflow Models have a scalability problem. As behavior gets more complex, the models become big. Tools often don't support modularization well.





Behavior: Mathematical

```
vector<int16, 3> aVector = [1; 2; 3] * 512;

vector<int16, 3> resultOfCrossProduct = aVector x aVector;

matrix<int16, 2x3> aMatrix = [1 + 2   2 * 7   42; 3   51   24];

matrix<int16, 3x2> transposedMatrix = aMatrix^T;
```

```
int32 averageIntArray(int32[] arr, int32 size) {
```

$$\text{return } \frac{\sum_{i=0}^{size} arr[i]}{size};$$

```
} averageIntArray (function)
```

Mathematical types, operators and notations can be used effectively to model behavior in some domains.

Enriched Type Systems and Contracts

```
section PowerUnits {
  exported unit W := kg·m²·s⁻²·s⁻¹ for Active power
  exported unit kW := <no spec> for Active power
  exported unit Var := W for Reactive power
  exported unit kVar := kW for Reactive power
  exported unit VA := W for Apparent power
  exported unit kVA := kW for Apparent power

  exported conversion W -> kW {
   val as <no type> -> val / 1000
  }
  exported conversion kW -> W {
   val as <no type> -> val * 1000
  }
} section PowerUnits
```

```
int16/m/ dAlt = cur->alt - prev->alt;
int8/s/ dTime = cur->time - prev->time;
Error: type int16 /s · m^(-1)/ is not a subtype of int16 /mps/
cur->vSpeed = dTime / dAlt;
```

The more semantics you put into models or code, the easier it is to implement meaningful checks.

```
exported cs interface IRTCAlarm {
  void setAlarm(uint8 day, uint8 hour,
    pre(0) (day >= 1 && day <= 31) || d
    pre(1) hour <= 23 || hour == RTC_NO
    pre(2) minute <= 59 || minute == RTC_NO_MINUTE_ALARM
  void setWeeklyAlarm(uint8 dayOfWeek, uint8 hour, uint8 minute, boolean repeat)
    pre(0) dayOfWeek <= 6
    pre(1) hour <= 23 || hour == RTC_NO_HOUR_ALARM
    pre(2) minute <= 59 || minute == RTC_NO_MINUTE_ALARM
  void clearAlarm()
}
```

Enriched Type Systems and Contracts

```
[Layer: ui]
CatalogUI                                   constraints
model   mbeddr.tutorial.main.newLanguages   imports     Catalog
package examples.layers                                 DatabaseStuff
                                                        PricingEngine

// The catalog port is supposed to have an error for demo purposes.
expo                                                    thing {
  requires IQuery catalog
  requires IPriceCalculator pricer
} component CatalogBrowser
```

Error: required interface must be from the same or a lower layer

The more semantics you put into models or code, the easier it is to implement meaningful checks.

Simple Better Abstractions

```
trysequentially {
  validateStep1(data, ...);
  validateStep2(data, ...);
  validateStep3(data, ...);
} on fail (errorcode) {
  handleFailedValidation(data, errorcode, ...);
}
```

Even small, local embedded DSLs can solve real problems. Anyone remember goto FAIL?

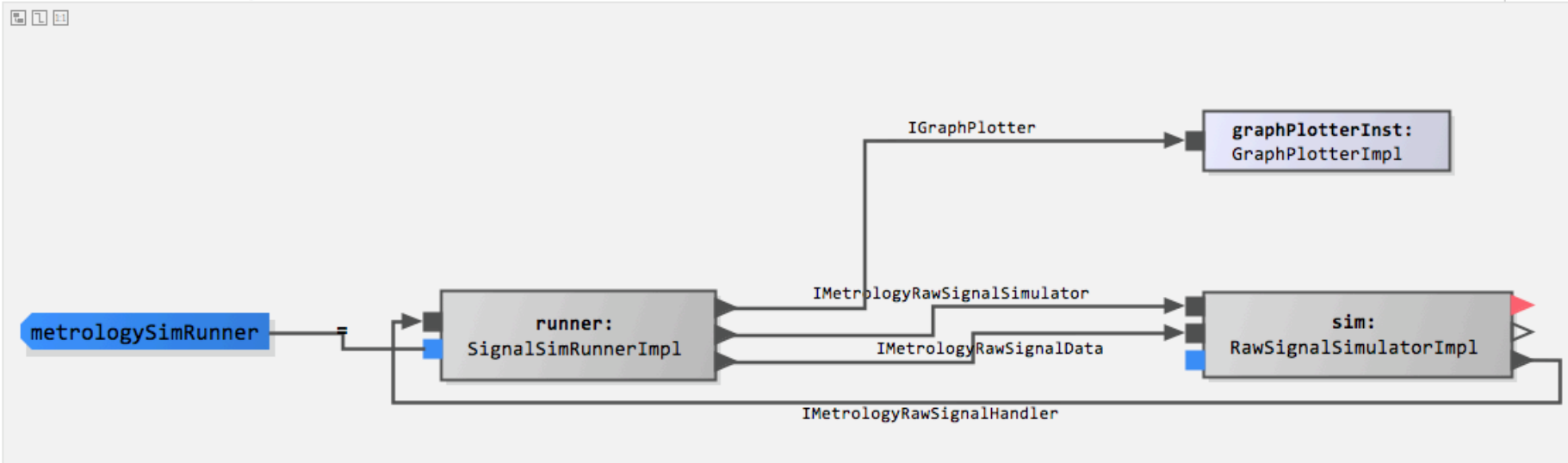
Modularization and Interfaces

```
exported component EraseFlashSegmentCommandHandlerImpl extends nothing {
  provides ICalibrationCommandHandler commandHandler
  requires IFlashMemoryController flashMemoryController

  uint8 commandHandler_getId() <= op commandHandler.getId {
    return CALIBRATION_COMMAND_ID_ERASE_FLASH_SEGMENT;
  } runnable commandHandler_getId

  CalibrationCommandStatus commandHandler_commandInvoked(Datagram* pRequestFrame,Datagram* pResponseFrame)
      <= op commandHandler.commandInvoked {
    uint16* pRequestData = ((uint16*) pRequestFrame->pVisiblePayload);

    // Read flash memory destination address from request message
    uint16* pFlashLocation = ((uint16*) *pRequestData);

    // Erase enclosing flash memory segment
    flashMemoryController.eraseSegment(pFlashLocation);

    // Set actual response data length
    pResponseFrame->visiblePayloadLength = 0;
    return CALIBRATION_COMMAND_STATUS_OK;
  } runnable commandHandler_commandInvoked
} component EraseFlashSegmentCommandHandlerImpl
```

Components, Ports and Interfaces are a well-established means of breaking down large systems.

Modularization and Interfaces

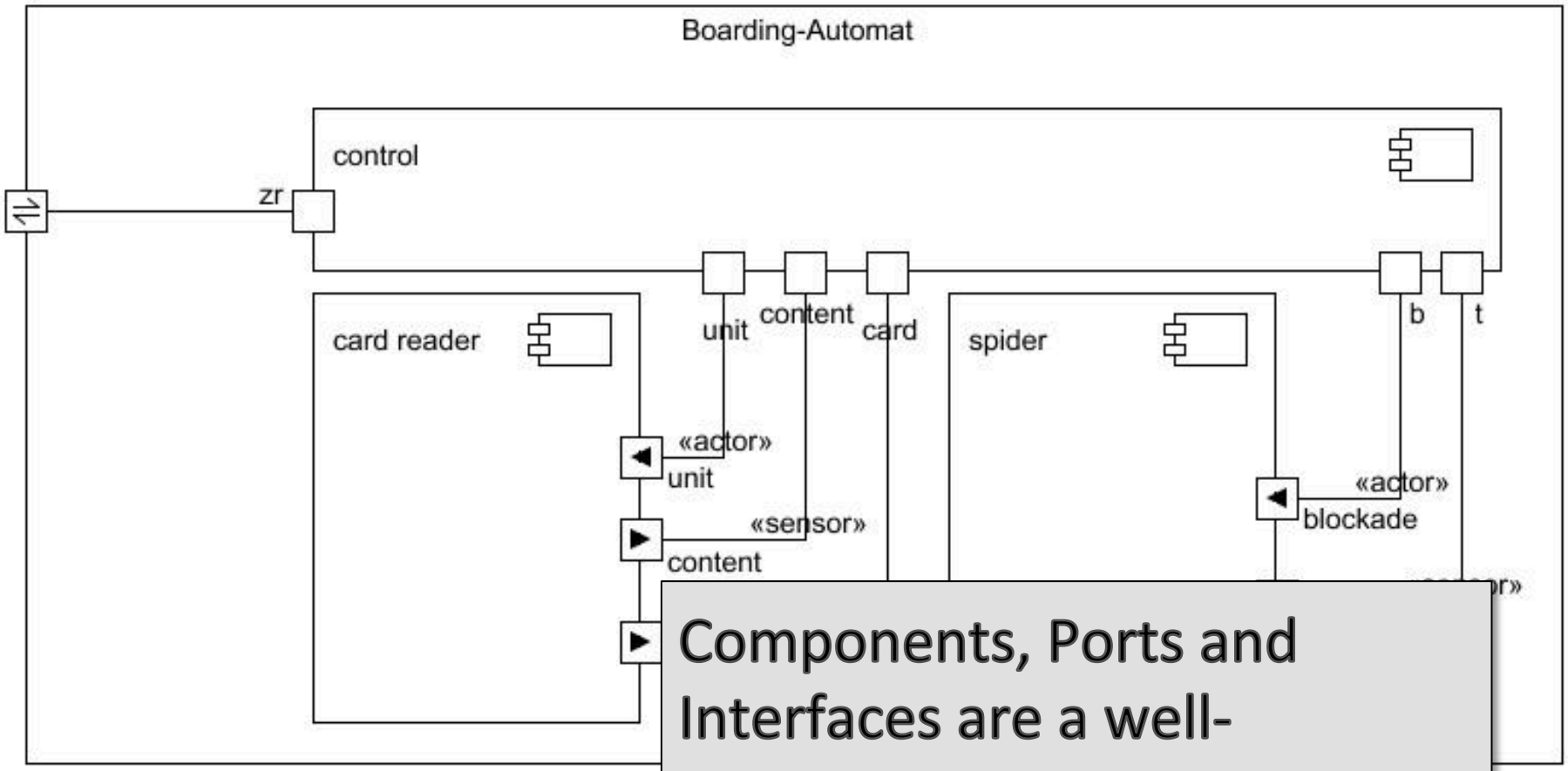
```
exported composite component MetrologyRawSignalSimulatorTestHarnessImpl {
  provides IMetrologyRawSignalSimulationRunner metrologySimRunner

  internal instances {
```




```
  }
}
```

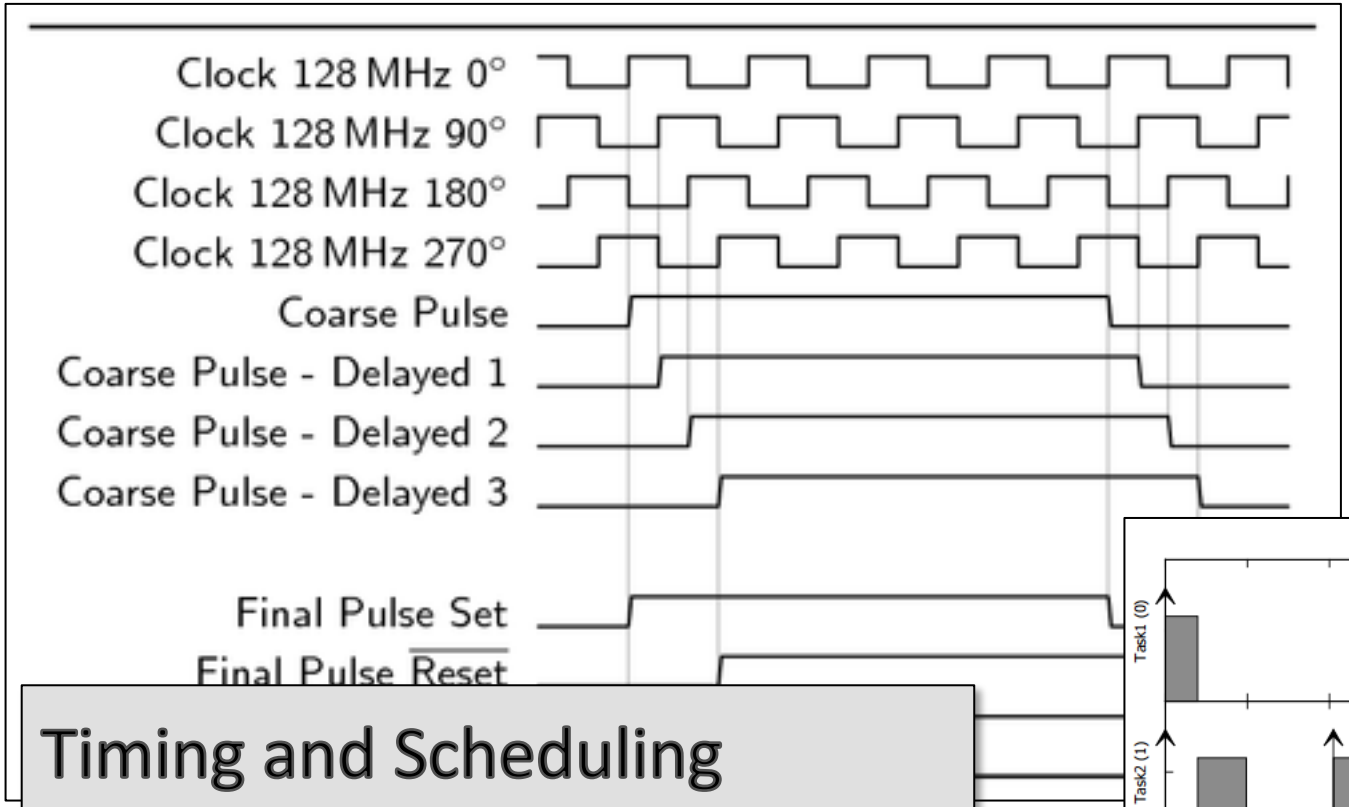
Components, Ports and Interfaces are a well-established means of breaking down large systems.

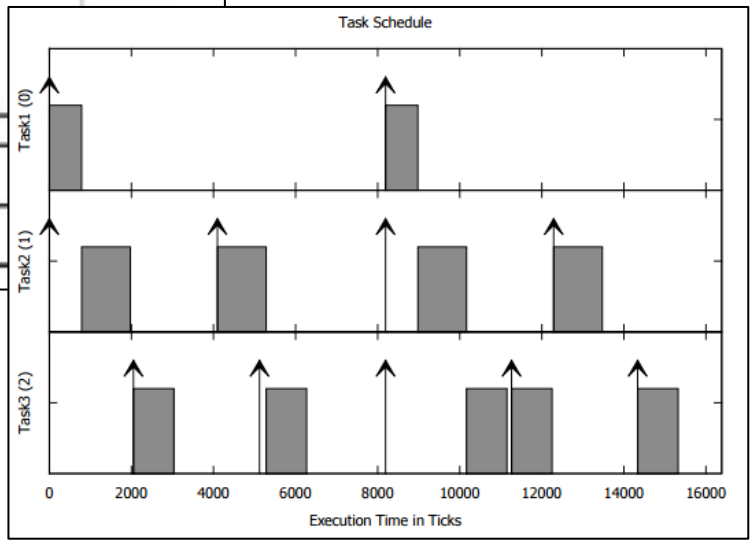
Modularization and Interfaces



Components, Ports and Interfaces are a well-established means of breaking down large systems.

Non-Functional: Timing





Timing and Scheduling Models are one of the ingredients for deployment onto hardware.

Non-Functional: Timing

OS Configutation and mapping to code made explicit – makes correct deployment simpler.

```
task Measurement {
  critical(data) {
    data.rpm = measureRPM();
    data.speed = measureSpeed();
  }
}

task Kombi {
  uint16 s = 0;
  uint16 r = 0;
  critical(data) {
    s = data.speed;
    r = data.rpm;
  }
  showInUI(s, r);
}
```

```
task Measurement prio = 1  wcet = 10 ms
task Kombi       prio = 2  wcet = 100 ms
memory layout {
  region ram:     0              .. 1024
  region eprom:   startOf(ram)   .. 2048
  region devices: endOf(eprom)   .. startOf(devices) + sizeOf(ram) * 2
}
```

Combinations

Embedding Procedural C Code or Math in Components, Dataflow Blocks, State Machine Actions...

```
simulink block: Integrator
exported atomicblock Integrator
    [double in                ]=>[double res]
    [double curTime           ]
    [optional boolean reset   ]
    [optional double resetIV  ]
  parameters [double initialValue;]
             [double minValue;    ]
             [double maxValue;    ]
  state      [double integrated = initialValue;]
             [double prevTime = -1;            ]
  on init    {
    integrated = initialValue;
    prevTime = 0;
  }
  ccode {
    output { res = ((integrated < minValue))?(minValue):(((integrated > maxValue))?(maxValue):(integrated)); }
    update {
      if (isConnected(reset)) {
        if (reset || prevTime < -0.5) {
          if (isConnected(resetIV)) {
            integrated = resetIV;
          } else { integrated = initialValue; }
        } if
      } else if (prevTime < -0.5) {
        integrated = initialValue;
      }
      if (prevTime >= -0.5) {
        integrated += in * (curTime - prevTime);
      } if
      prevTime = curTime;
    }
  };
```

Combinations

Embedding Procedural C Code or Math in Components, Dataflow Blocks, State Machine Actions...

Embedding a State Machine in a Dataflow Block

Using State Machines to Control Dataflow Models.

Using Expressions in Contracts (DF, SM, Comp)

All kinds of DSLs in other languages (DF, SM, Comp)

5

Performance and Resources

Generated Code is

slow, ugly and bloated.

Or is it?

SPIRAL: A Generator for Platform-Adapted Libraries of Signal Processing Algorithms

Markus Püschel* Bryan Singer Jianxin Xiong José M. F. Moura
Jeremy Johnson David Padua Manuela Veloso Robert W. Johnson

Abstract

SPIRAL is a generator of libraries of fast software implementations of linear signal processing transforms. These libraries are adapted to the computing platform and can be re-optimized as the hardware is

... empirical data that demonstrates the high performance ...

L: the mathematical frame-
the formula generator that
articular signal processing
f freedom when translating
elligent search engine that
finds within the large space of alternative formulas and implementations the "best" match to the given computing platform. We present empirical data that demonstrates the high performance of SPIRAL generated code.

Proceedings of the IEEE
http://spiral.ece.cmu.edu:8080/pub-spiral/pubfile/spiral_1.pdf

Software Synthesis and Code Generation for Signal Processing Systems

Shuvra S. Bhattacharyya, *Member, IEEE*, Rainer Leupers, and Peter Marwedel, *Member, IEEE*

***Abstract*—The role of software is becoming increasingly important in the implementation of digital signal processing (DSP) applications. As this trend intensifies, and the complexity of applications escalates, we are seeing an increased need for automated tools to aid in the development of DSP software. This paper reviews the state-of-the-art in programming language and compiler technology for DSP software implementation. In particular, we review techniques for high-level block-diagram-based modeling of DSP applications; the translation of block-diagram specifications into efficient C programs using global target-independent optimization techniques; and the compilation of C programs into streamlined machine code for programmable DSP processors using architecture-specific and retargetable back-end optimizations. We also point out important directions for further investigation.**

***Index Terms*—**
targetable compil

take significantly less verification effort compared to custom hardware solutions.

Although the flexibility offered by software is critical in DSP applications, the implementation of production quality DSP software is an extremely complex task. The complexity arises from the diversity of critical constraints that must be satisfied; typically, these constraints involve stringent requirements on metrics, such as latency, throughput, power consumption, code size, and data storage requirements. Additional constraints include the need to ensure key implementation properties, such as bounded memory requirements and deadlock-free operation.

eral-pur-
engage
de at the

... into efficient C programs using global target-dependent optimizations ...

IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing

http://www.ece.umd.edu/DSPCAD/papers/bhat2000x6.pdf

Using C Language Extensions for Developing Embedded Software: A Case Study

Markus Voelter
independent/itemis, Germany
voelter@acm.org

Arie van Deursen
Delft University of Technology, The Netherlands
Arie.vanDeursen@tudelft.nl

Bernd Kolb, Stephan Eberle
itemis AG, Germany
{kolb|eberle}@itemis.de

Abstract

We report on an industrial ca
embedded software for a smart
ming language and domain-spe
components, physical units, sta
terrupts. We find that the exten
managing the complexity of t
testability mainly by supporting
ing, as illustrated by low integration efforts. The extensions also do not incur significant overhead regarding memory consumption and performance. Our case study relies on mbeddr, an extensible version of C. mbeddr, in turn, builds on the MPS language workbench which supports modular extension of languages and IDEs.

... do not incur significant overhead regarding memory consumption and performance.

hat extent
beddr are
opment of
lyze, in a
omplexity,
ftware, as

Audience We target language engineering researchers (interested in empirical data justifying their work or looking to understand problems they may be able to solve) as well as embedded systems developers (seeking to understand how language extensions can help them in practice).

Categories and Subject Descriptors D.3.2 [*Extensible languages*]; D.3.4 [*Code Generation*]; D.2.3 [*Program Editors*]; C.3 [*Real-time and embedded systems*]

Keywords Embedded Software, Language Engineering, Language Extension, Domain-Specific Language, Case Study

OOPSLA 2015

http://voelter.de/data/pub/mbeddr-cs-oopsla2015-preprint.pdf

Generated Code is

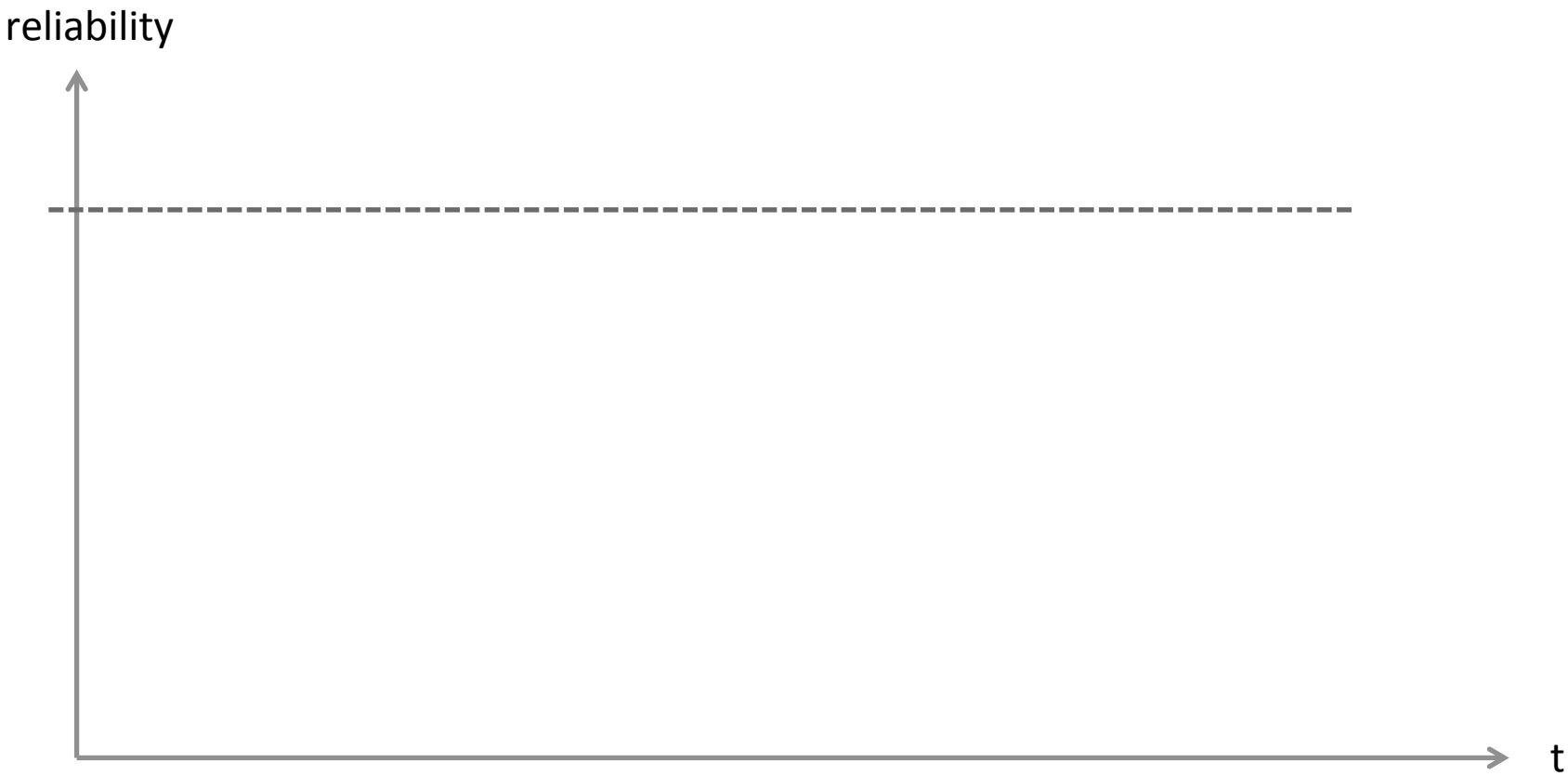
slow, ugly and bloated.

It is not, if you do it right.

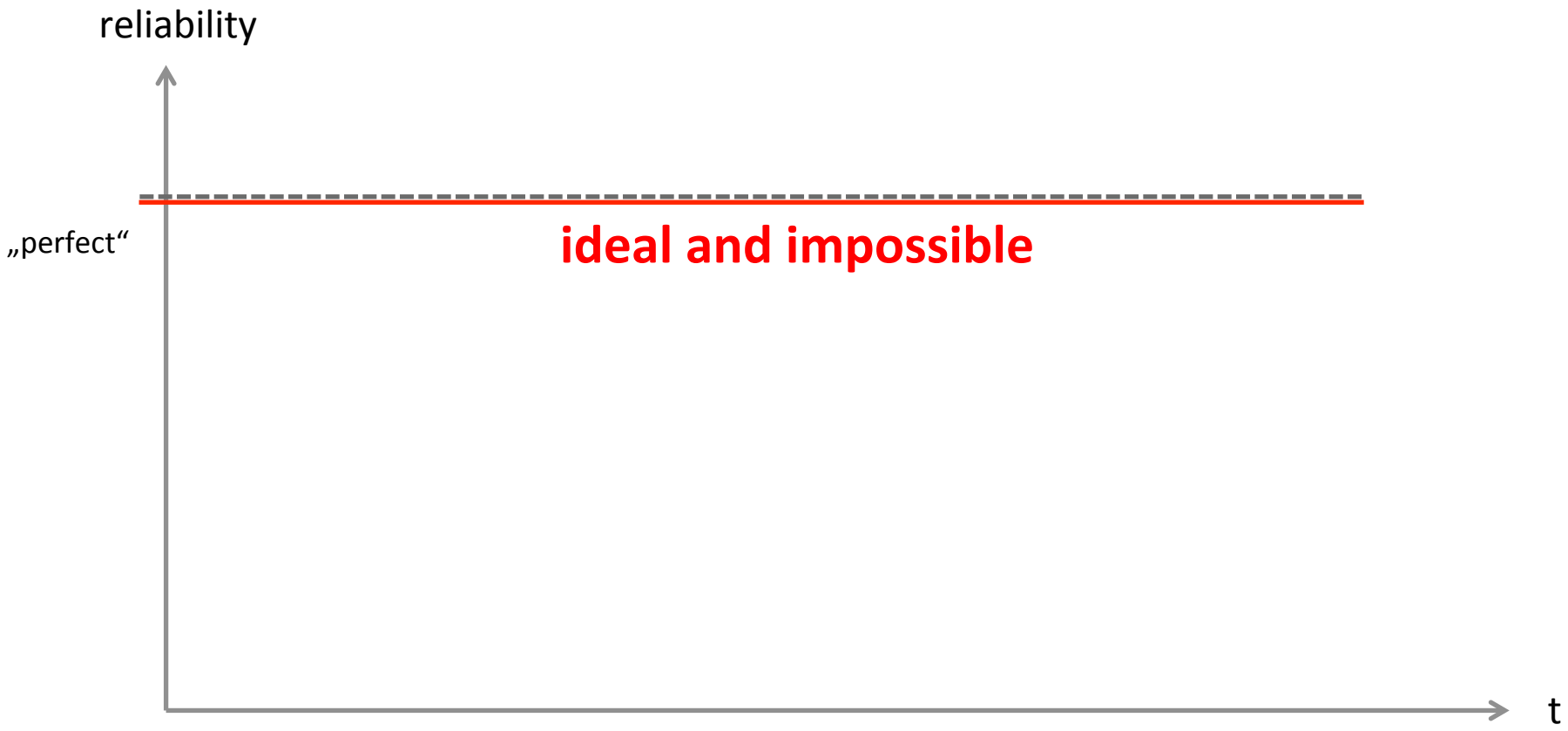
7

Reliability

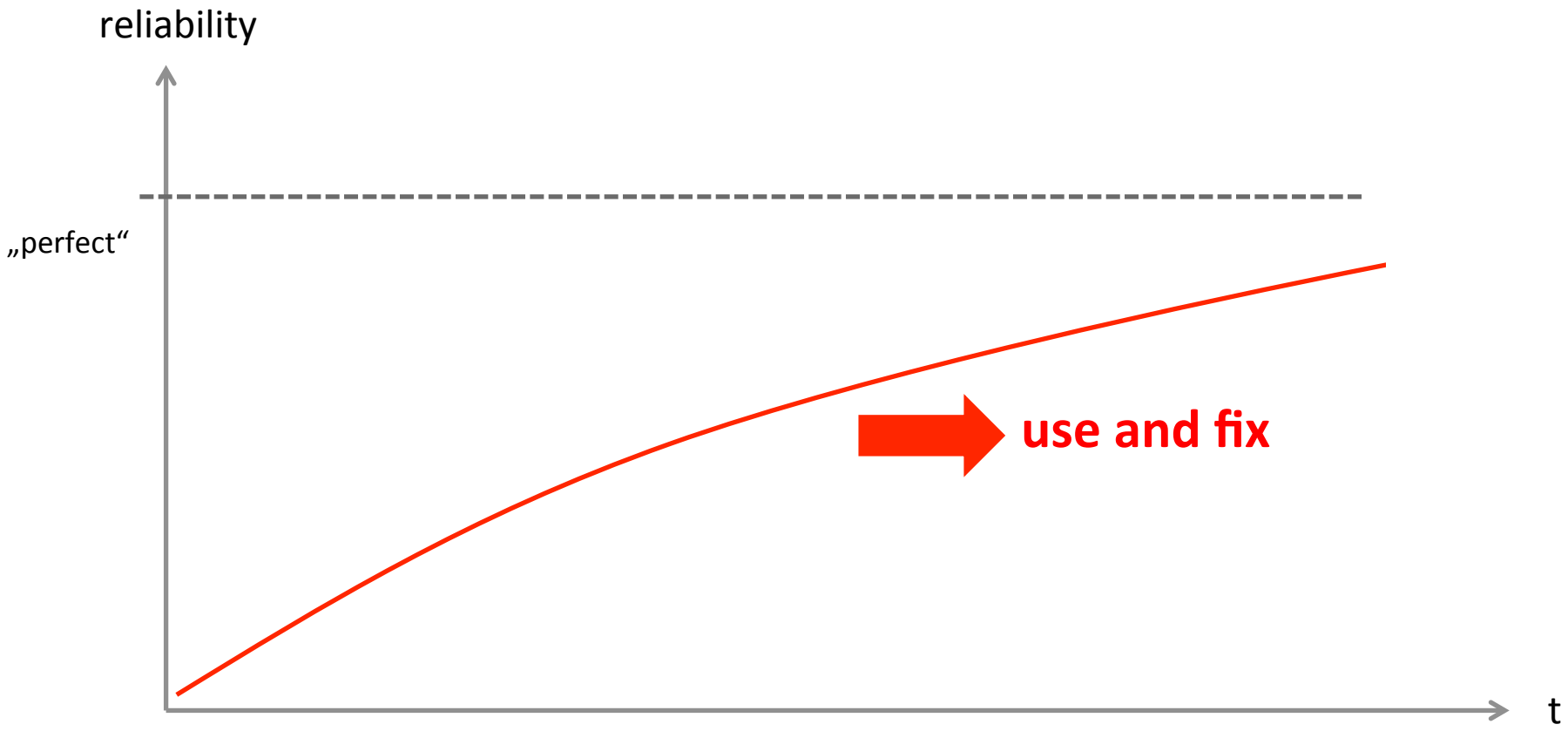
Errors over Time



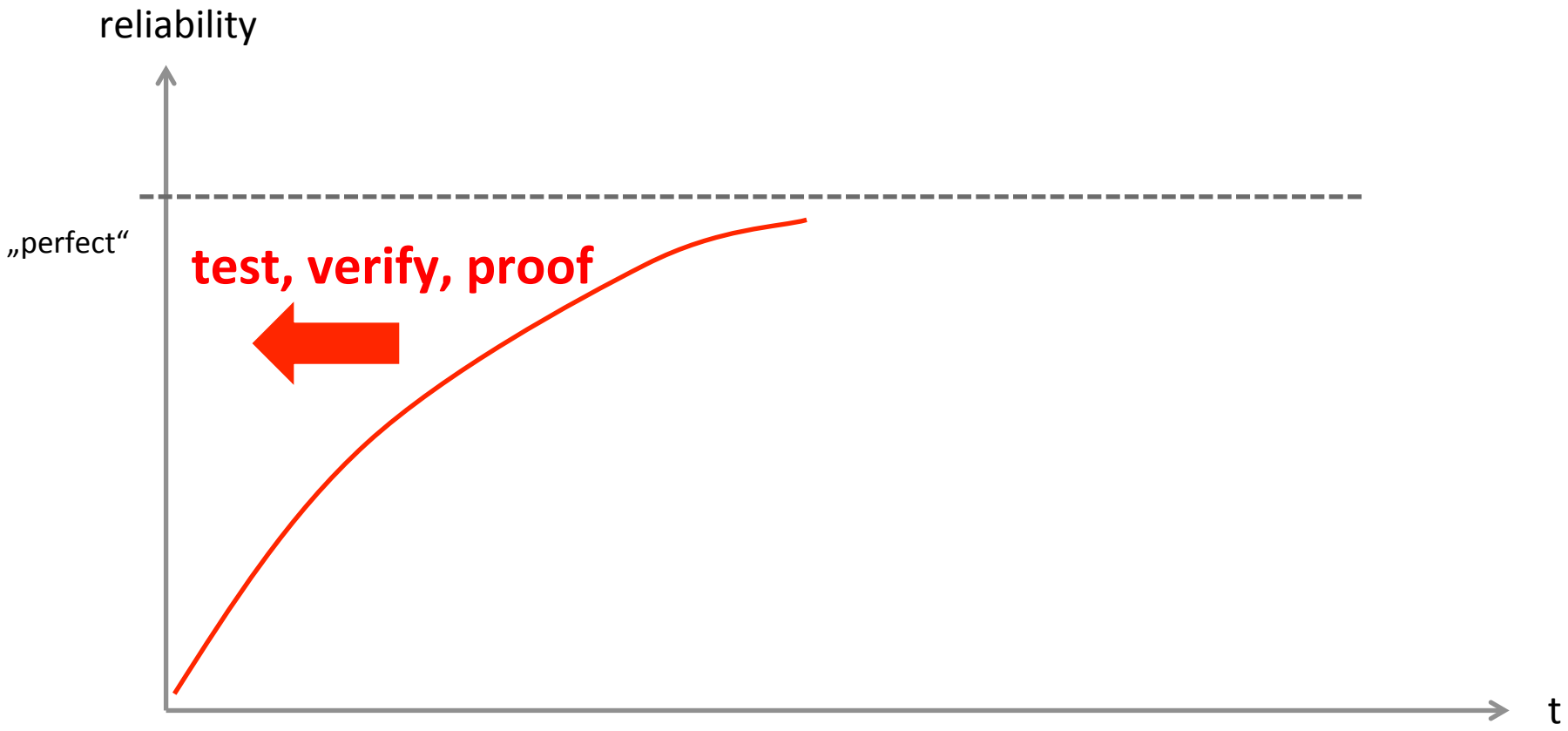
Errors over Time



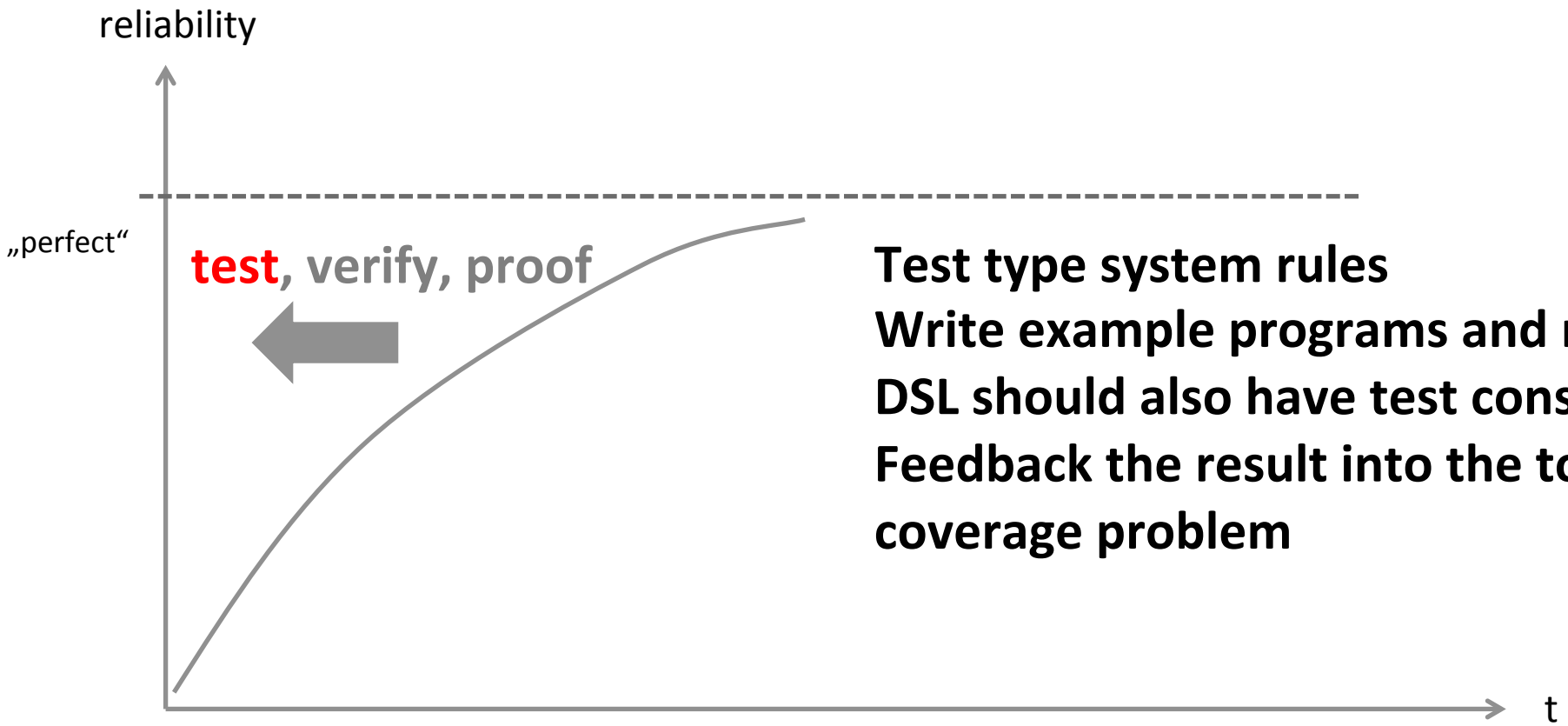
Errors over Time



Errors over Time

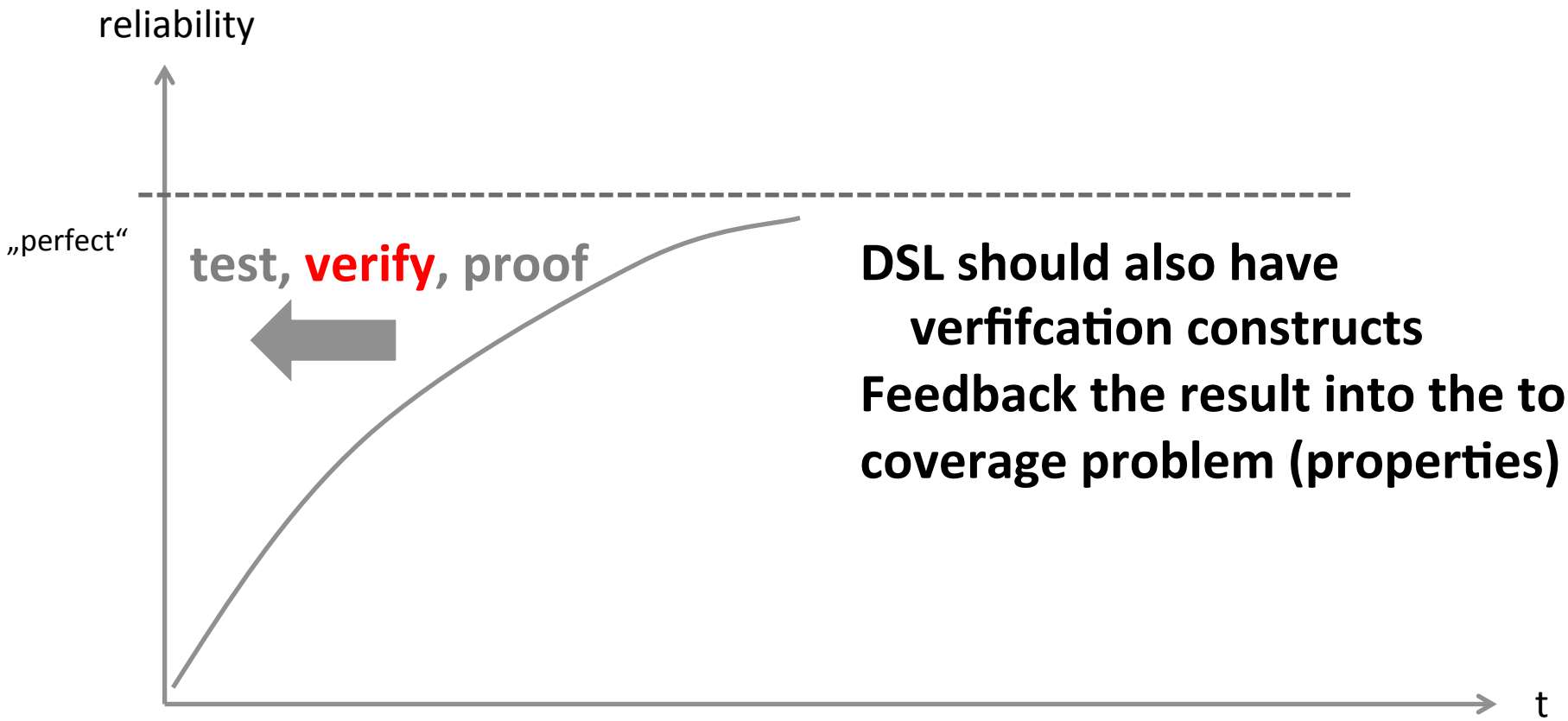


Errors over Time



Test type system rules
Write example programs and run
DSL should also have test constru
Feedback the result into the tool
coverage problem

Errors over Time



DSL should also have verfifcation constructs

Feedback the result into the tool coverage problem (properties)

Automated Domain-Specific C Verification with mbeddr

Zaur Molotnikov
Fortiss Institute
Guerickestraße 25
Munich, Germany
molotnikov@fortiss.org

Markus Völter
independent/itemis
Oetztaler Straße 38
Stuttgart, Germany
voelter@acm.org

Daniel Ratiu
Fortiss Institute
Guerickestraße 25
Munich, Germany
ratiu@fortiss.org

ABSTRACT

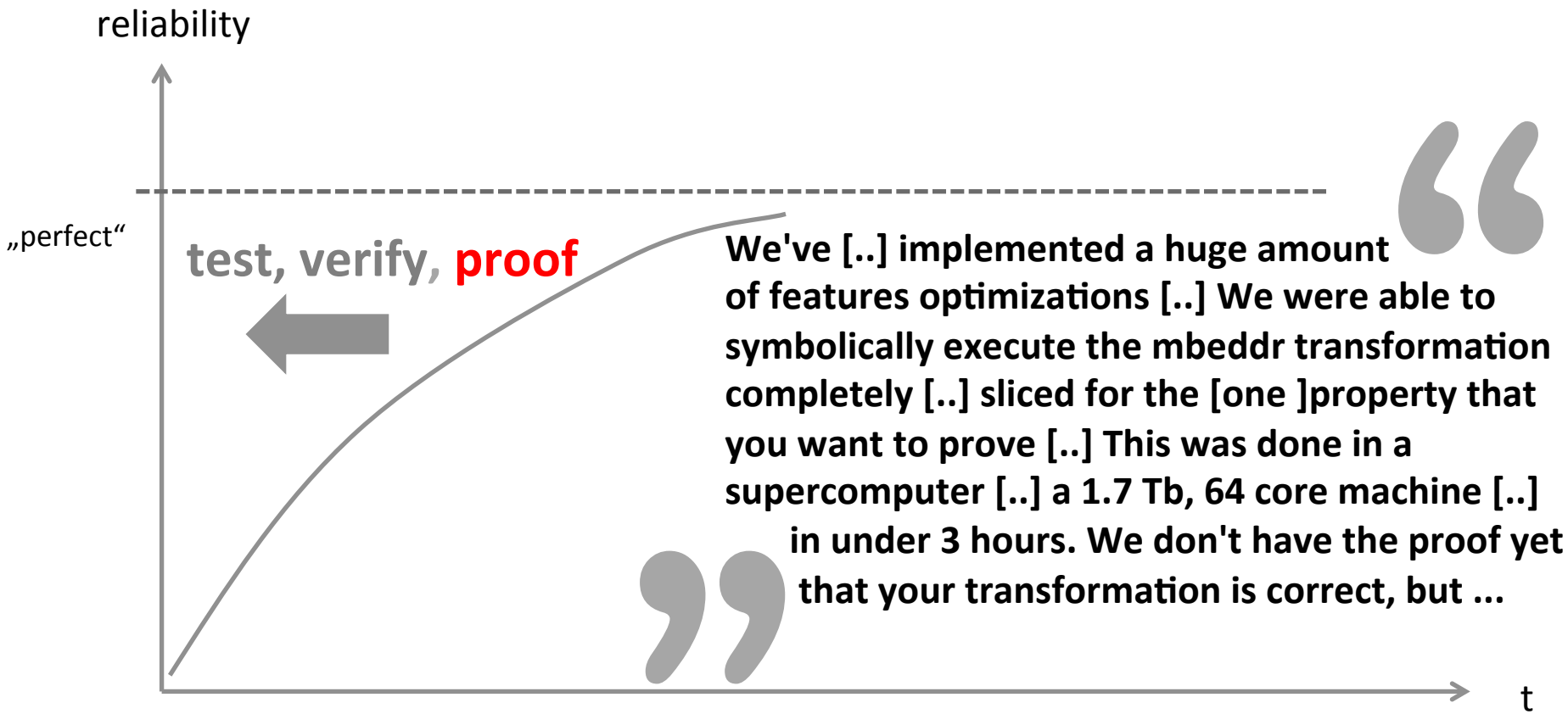
When verifying C code, two major problems must be addressed. One is the specification of the verified systems properties, the other one is the construction of the verification environment. Neither C itself, nor existing C verification tools, offer the means to efficiently specify application domain-level properties and environments for verification. These two shortcomings hamper the usability of C verification, and limit its adoption in practice. In this paper we introduce an approach that addresses both problems and results in user-friendly and practically usable C verification. The novelty of the approach is the combination of domain-specific language engineering and C verification. We apply the approach in the domain of state-based software, using mbeddr and CBMC. We validate the implementation with an example from the Pacemaker Challenge, developing a functionally verified, lightweight, and deployable cardiac pulse generator. The approach itself is domain-independent.

Requirements such as this one describe the properties the system should have. To make them verifiable by tools, they have to be expressed as formal verification conditions. A (not necessarily correct) implementation of this pacing logic is given in Listing 1. This code could be verified using C verification tools such as CBMC [10], SATABS [11] or CPAchecker [7]. Working at the abstraction level of C, they can be used to, for example, check assertions or error-label reachability. This makes it impractical to directly represent the application domain-level semantics implied by requirements such as those described above [13, 30].

```
int t = 0;
bool makePace(Event e) {
  switch(e) {
    case Tick:
      if (t < LRI) {
        ++t;
        return false;
      } else {
        t = 0;
        return true;
      }
    case Sense:
      if (t < VRP) {
        return false;
      } else {
        t = 0;
        return false;
      }
    }
  }
```

Listing 1: Sample pacing logic implementation in C

Errors over Time



We've [..] implemented a huge amount of features optimizations [..] We were able to symbolically execute the mbeddr transformation completely [..] sliced for the [one]property that you want to prove [..] This was done in a supercomputer [..] a 1.7 Tb, 64 core machine [..] in under 3 hours. We don't have the proof yet that your transformation is correct, but ...

Proving a non-trivial language/tool correct is MAJOR work.

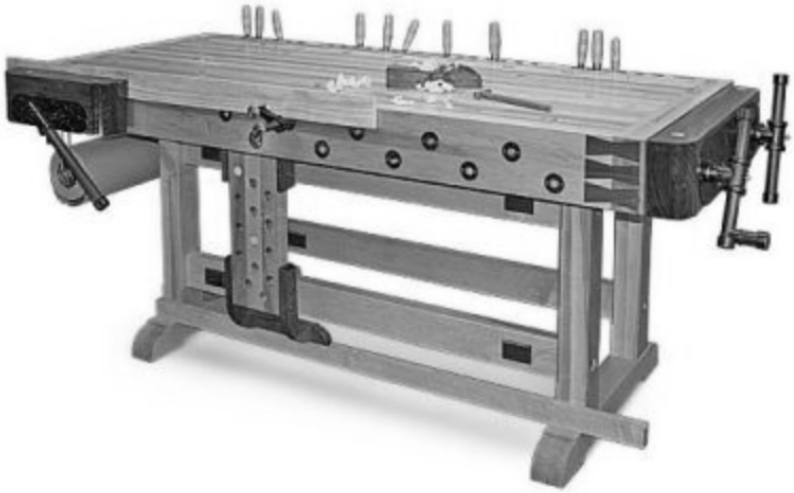
6

Your own!

Language Workbenches

Language Workbench

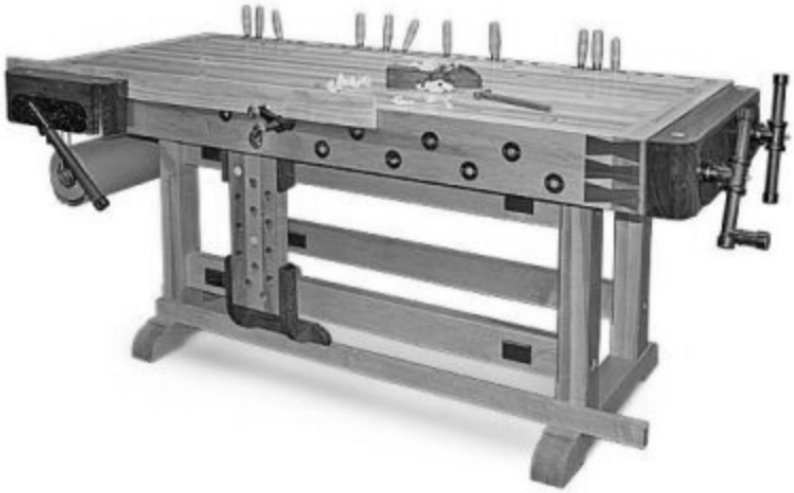
(Martin Fowler, 2004)



Freely
define
languages and
integrate
them

Language Workbench

(Martin Fowler, 2004)



powerful

editing

testing

refactoring

debugging

groupware

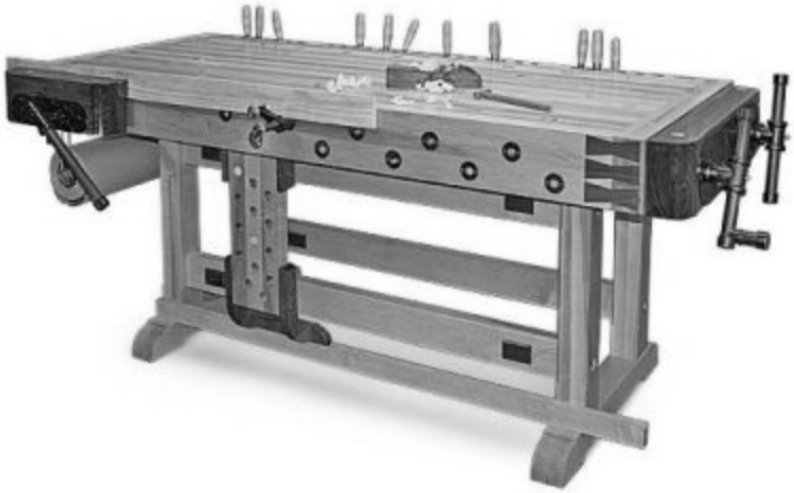
language definition

implies

IDE definition

Language Workbench

(Martin Fowler, 2004)

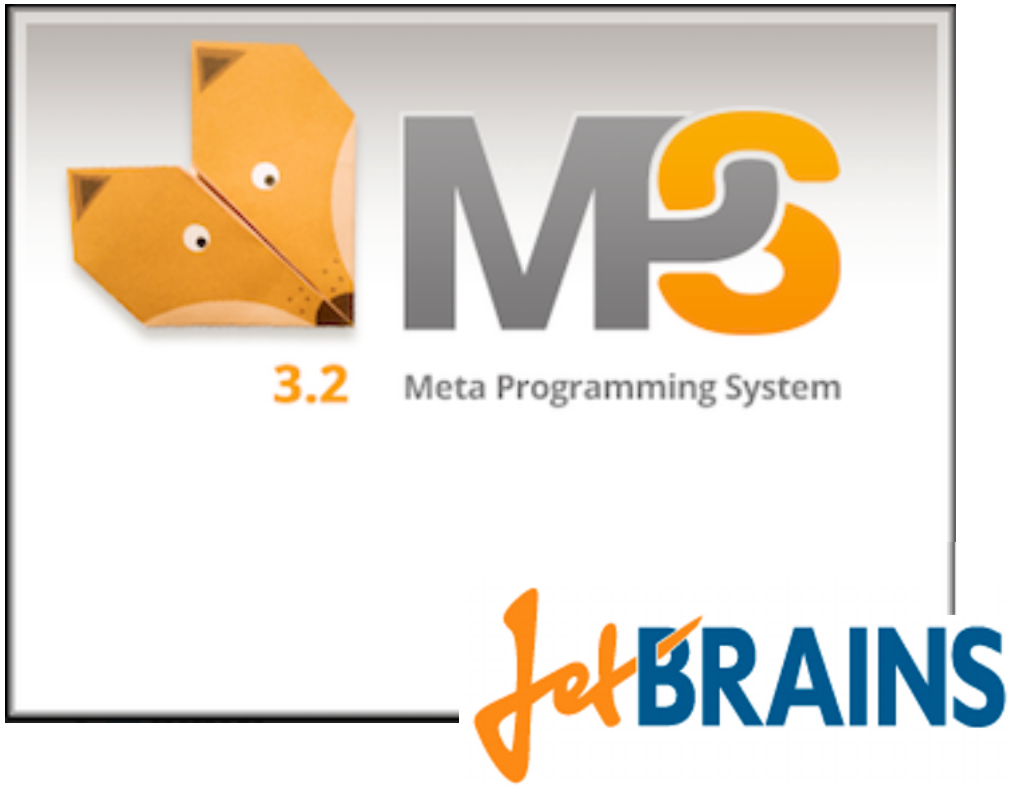


support for „classical“ programming + „classical“ and modeling

There‘s no difference!

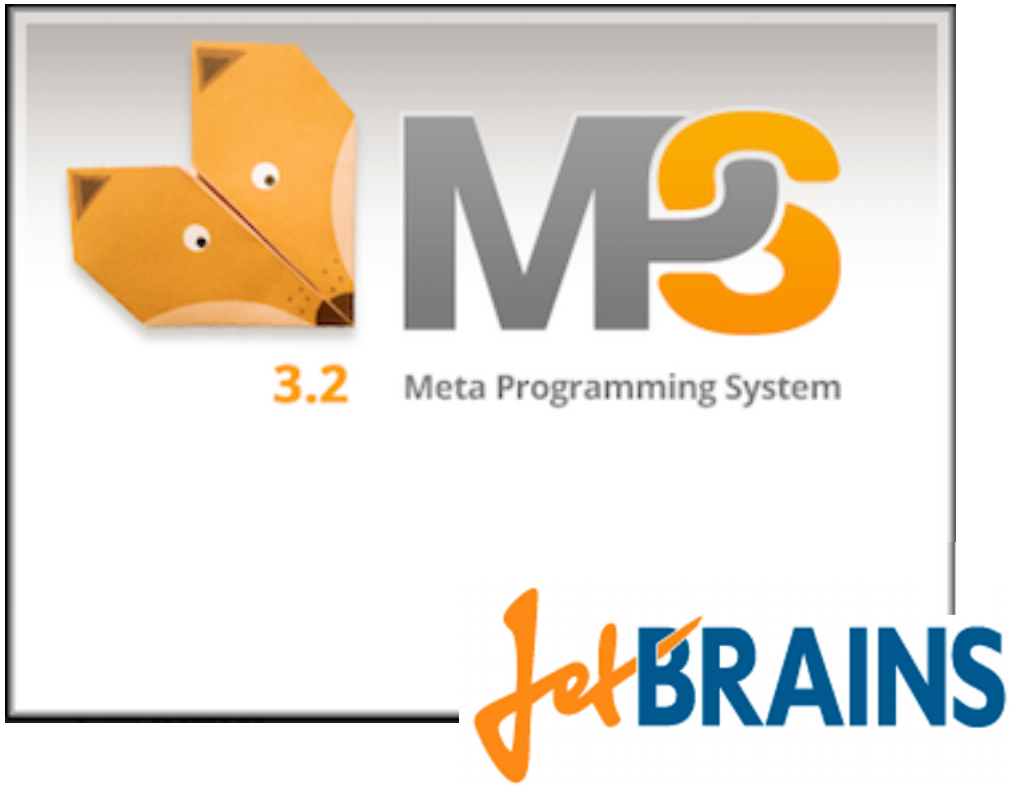
8

JetBrains MPS

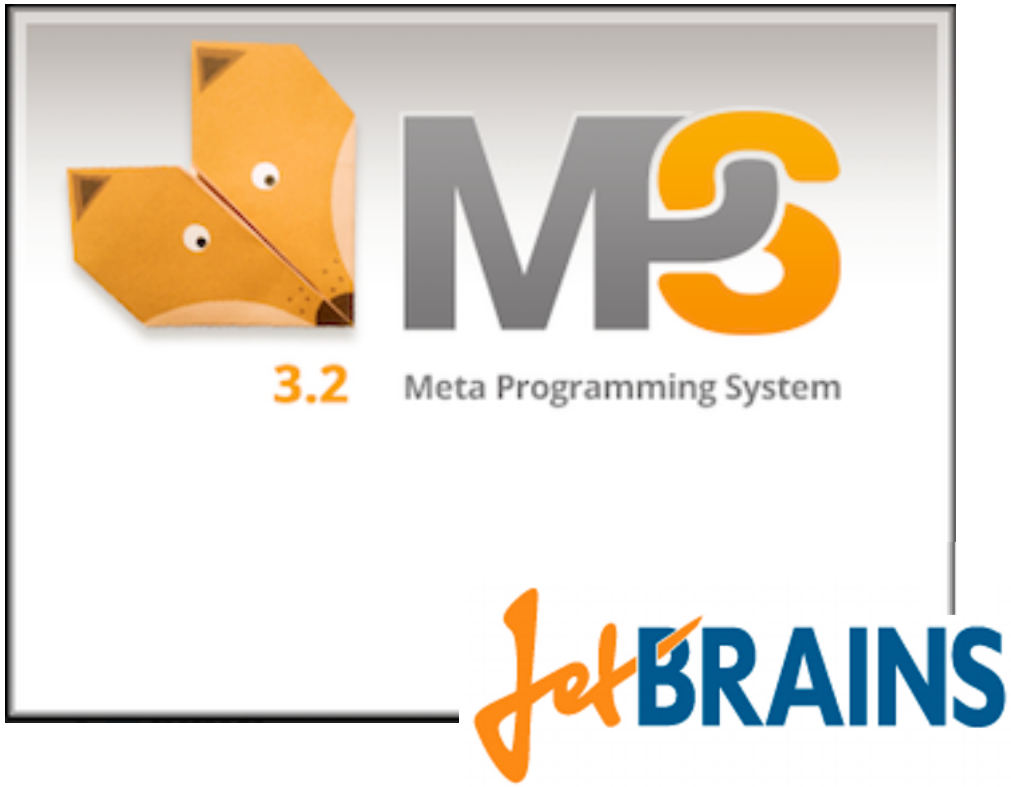


A Language Workbench –

a tool for defining, composing and using ecosystems of languages.



Open Source
Apache 2.0
http://jetbrains.com/mps

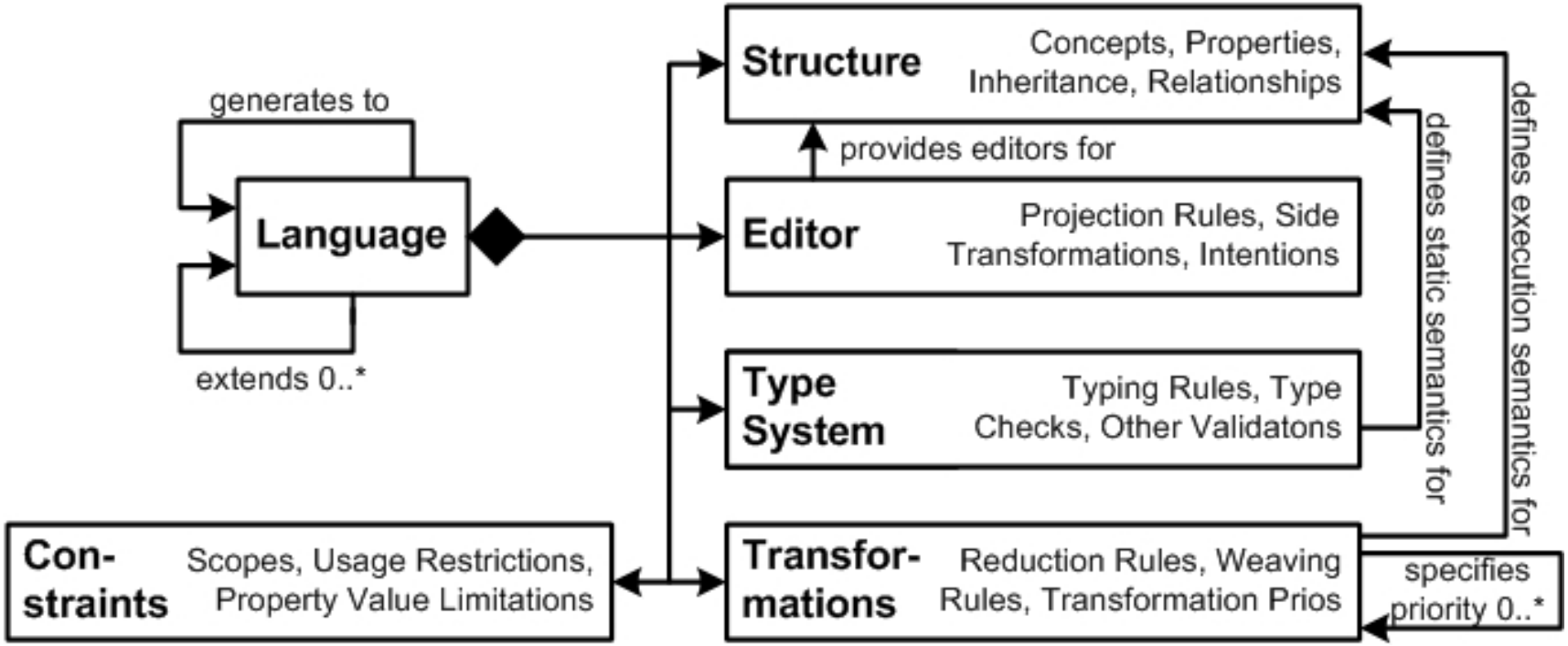


V 3.2.3 is current.

V 3.3 currently developed

[Language Workbench]

Comprehensive Support for many aspects of Language Definition.



+ Refactorings, Find Usages, Syntax Coloring, Debugging, ...

DLT645ProtocolImpl

DLT645ProtocolImpl ▶ handleDataLength

DLT645ProtocolImpl ▷ DLT645RxFrameParser

Events

dataReceived(uint8 data)

States

```
idle
  [data == DLT645_FRONT_LEADING_BYTE] -> awaking
  { }
awaking
  [data + DLT645_FRAME_STARTFLAG] -> receivingFrame
  {
    send startFrame(idx, data);
    idx++;
  }
receivingFrame
  [idx < length - 1 && idx != DLT645_HEADER_DATALENG
    -> receivingFrame
  {
    send frameByte(idx, data);
    idx++;
  }
  [idx == DLT645_HEADER_DATALENGTH_OFFSET] -> recei
  {
    send dataLength(idx, data);
    idx++;
    length = DLT645_FR + DLT645_TRAILE
        data;
  }
  [idx == lengt
```

Error: type boolean is not a subtype of uint8

DLT645_FRAME_ENDFLAG ^DLT645ProtocolDefinitions.DLT645_FRAME_ENDFLAG (GlobalConstantDeclaration)
DLT645_FRAME_STARTFLAG ^DLT645ProtocolDefinitions.DLT645_FRAME_STARTFLAG (GlobalConstantDeclaration)
DLT645_FRONT_LEADING_BYTE ^DLT645ProtocolDefinitions.DLT645_FRONT_LEADING_BYTE (GlobalConstantDeclaration)
DLT645_FRONT_LEADING_BYTE_COUNT ^DLT645ProtocolDefinitions.DLT645_FRONT_LEADING_BYTE_COUNT (GlobalConstantDeclaration)

UnitDeclarations

FractionUnits ▶ nounit -> %

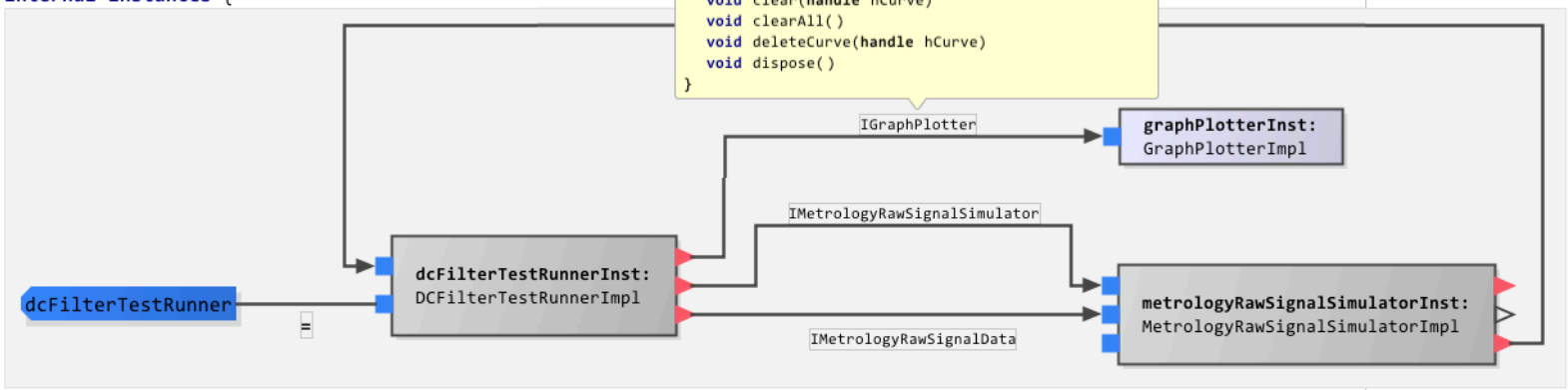
```
UnitDeclarations
model   com.itemis.embedded.platform.implementation.intf.units
imports

section FractionUnits {
  exported unit % := <no spec> for Fraction
  exported unit ‰ := <no spec> for Fraction
  exported unit ppm := <no spec> for Fraction

  exported conversion nounit -> % {
   val as <no type> -> val * 100
  }
  exported conversion % -> nounit {
   val as <no type> -> val / 100
  }

  exported conversion nounit -> ‰ {
   val as <no type> -> val * 1000
  }
```

DCFilterTestHarness

DCFilterTestRunnerImpl ▶ dcFilterTestRunner_dispose

```
  } runnable dcFilterTestRunner_dispose
} component DCFilterTestRunnerImpl

exported composite component DCFilterTestHarnessImpl {
  provides IDCFilterTestRunner dcFilterTestRunner

  internal instances {
```

```
exported cs interface IGraphPlotter {
  handle newCurve(Color color, string const label)
  void addPoint(handle hCurve, double x, double y)
  void plot(string const title, handle hCurve)
  void plotAll(string const title)
  void clear(handle hCurve)
  void clearAll()
  void deleteCurve(handle hCurve)
  void dispose()
}
```




```
  }
}
```

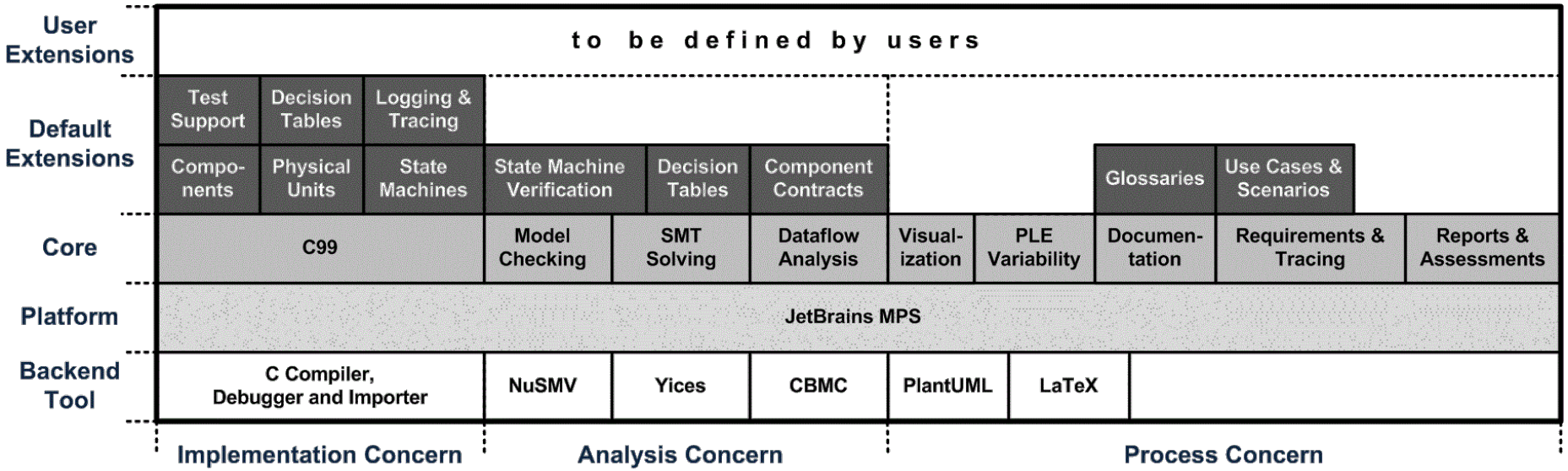
Diagram Palette

Event Log 3 | 2: Inspector

9

mbeddr

An extensible set of integrated languages for embedded software engineering.



BMW Car IT

Open Source @ eclipse.org
Eclipse Public License 1.0
http://mbeddr.com

Bundesministerium
für Bildung
und Forschung

mbeddr developers

itemis — 7 developers, project management

fortiss — 2 developers, verification stuff

SIEMENS — 1 developer, verification stuff

SIOUX — 3 developers, C++

---

strategic collaboration with JetBRAINS

Smart Meter

ca. 60,000 SLOC

Great modularity
(due to components)

Great Test Coverage (80%)

Physical Units Detect Errors

Custom Extensions: Messages, Registers, Interrupts

Much Lower Integration Effort

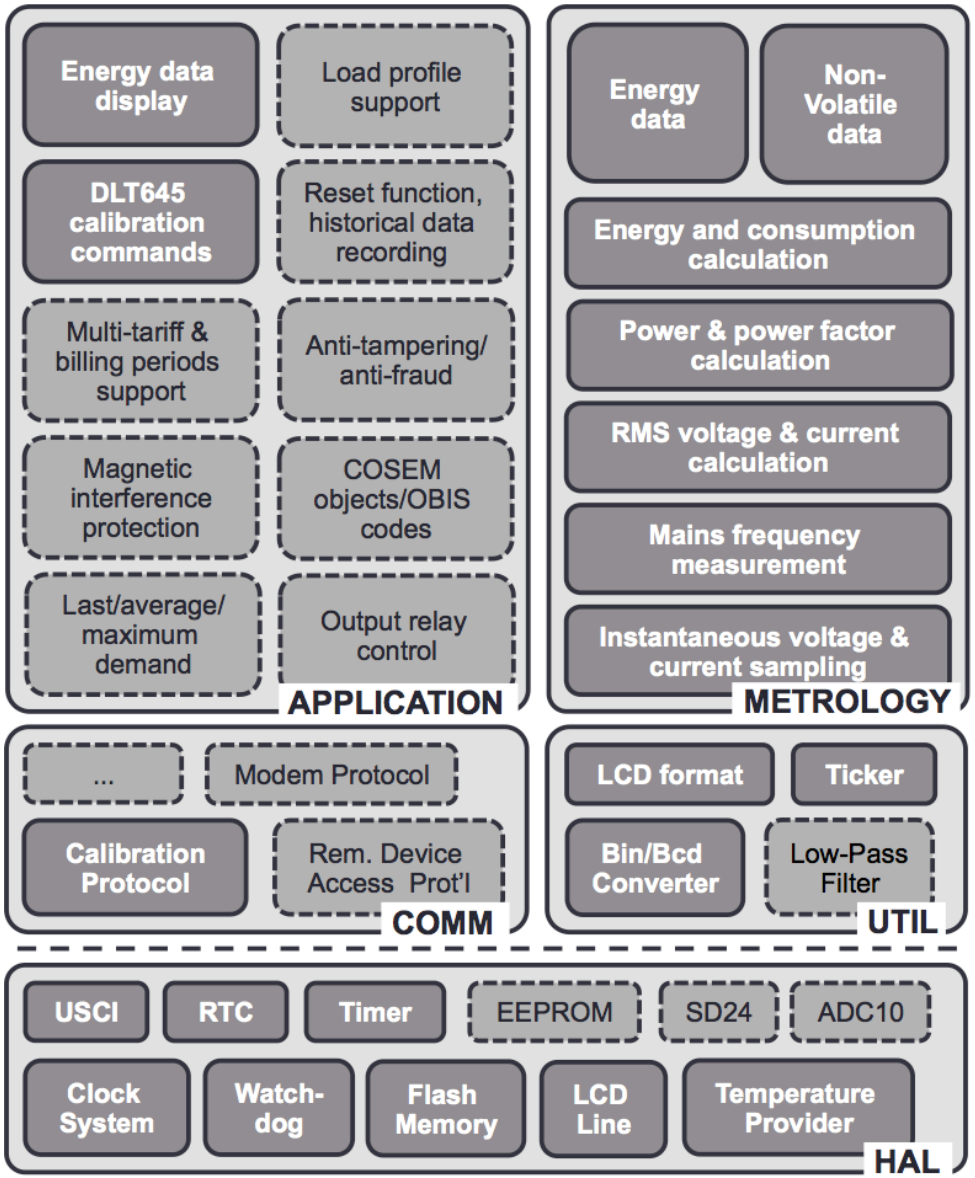
Smart Meter Key Drivers

Mastering complexity through well structured, modular architecture

Reuse through platform-based development approach

Hardware-independent testing

Incremental integration and commissioning



Embedded Software Designer

LMS INTERNATIONAL
Researchpark Haasrode 1237 | Interleuvenlaan 68 | B-3001 Leuven [Belgium]
T +32 16 384 200 | F +32 16 384 350 | info@lmsintl.com | www.lmsintl.com

Worldwide
For the address of your local representative, please visit www.lmsintl.com/lmsworldwide

LMS is a leading provider of test and mechatronic simulation software and engineering services in the automotive, aerospace and other advanced manufacturing industries. As a business segment within Siemens PLM Software, LMS provides a unique portfolio of products and services for manufacturing companies to manage the complexities of tomorrow's product development by incorporating model-based mechatronic simulation and advanced testing in the product development process. LMS tunes into mission-critical engineering attributes, ranging from system dynamics, structural integrity and sound quality to durability, safety and power consumption. With multi-domain and mechatronic simulation solutions, LMS addresses the complex engineering challenges associated with intelligent system design and model-based systems engineering. Thanks to its technology and more than 1250 dedicated people, LMS has become the partner of choice of more than 5000 manufacturing companies worldwide. LMS operates in more than 30 key locations around the world.

Siemens PLM Software

SIEMENS

LMS®

A Siemens Business

10

Recap/
Big Picture

Modeling and Model-Driven *
is a diverse field.

Modeling is more than boxes + lines.

It can be a huge benefit!

BYO language is an option!

THANK YOU!

voelter@acm.org
www.voelter.de
@markusvoelter