AE3212-II - Simulation, Verification and Validation

Wouter van der Wal

11-2-2020

Staff

Wouter van der Wal
(Course coordinator)

w.vanderwal@tudelft.nl

Julien van Campen
(Coordinator, structural analysis)

Alexander in't Veld
(Flight Dynamics)

Hans Mulder
(Flight Test)

+ 12 TA's, PhD students and staff from C&S and ASM

Contents for today

- 13:45-14:30
  Introduction to simulation, verification and validation
  Software verification
  (Wouter van der Wal)

- 14:45-15:30
  Structural Analysis Theory and Structural Analysis Assignment
  (Julien van Campen)

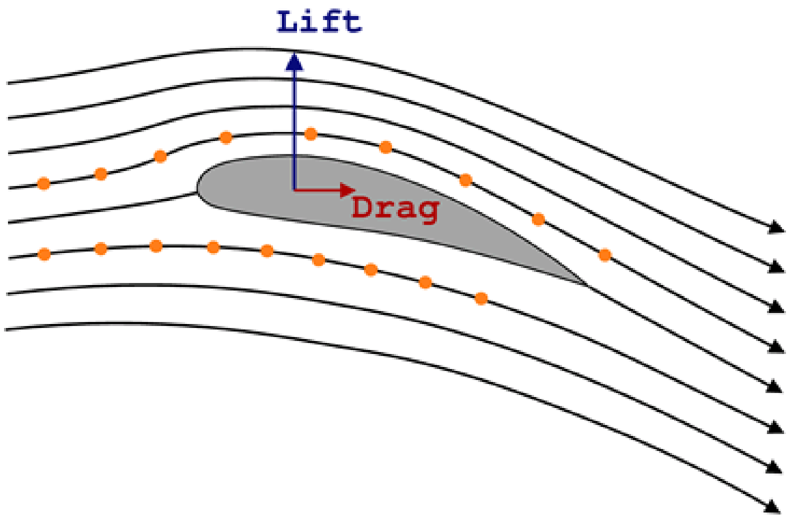
- 15:45-16:30
  Case study + Rules & Guidelines
  (Wouter van der Wal)

What is simulation?

"develop a (mathematical / numerical) model of the physical model of a physical problem and generate results"

BSc program

- Statics
- Dynamics
- Thermodynamics
- Waves and electromagnetism
- Aerodynamics
- Structural analysis and design
- Vibrations
- **Experimental research and data analysis**
- Flight and orbital mechanics
- Computational modelling
- + tools (mathematics, programming)



Imagine you work for a small airplane manufacturer and you have developed an autopilot.

Would you fly in the airplane and let the autopilot land the airplane?

Would you fly in the airplane and let the autopilot land the airplane when your colleague programmed the autopilot?

What could go wrong?

ExoMars, October 2016

"an unexplained saturation of its inertial measurement unit, which delivered bad data to the lander's computer and forced a premature release of its parachute."

Spacenews.com

What is Verification?

To determine if a simulation model accurately represents the chosen physical model

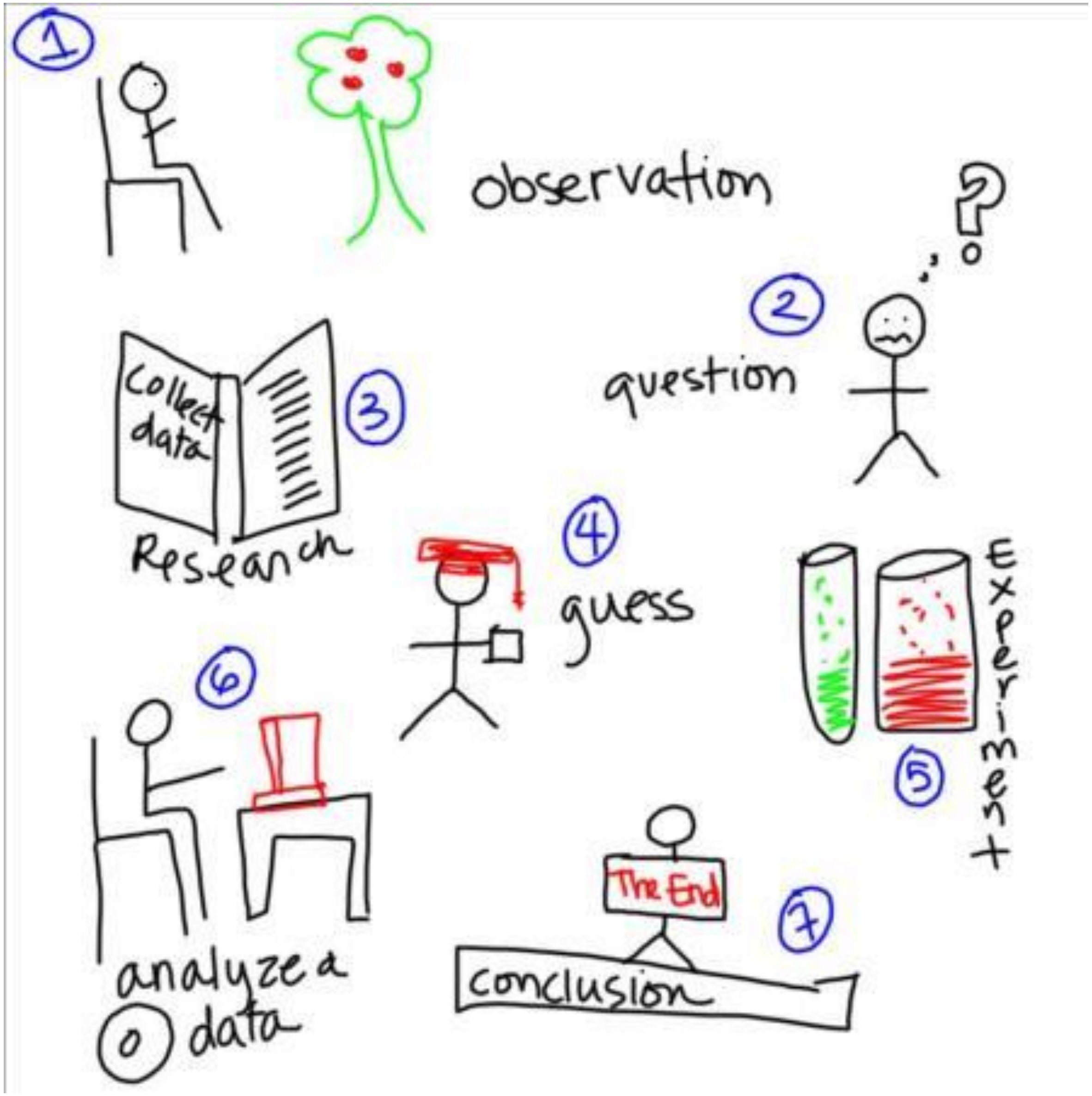
"Verification proves that a realized product for any system model within the system structure conforms to the build-to requirements (for software elements) or realize-to specifications and design descriptive documents (for hardware elements, manual procedures…)."

NASA system engineering handbook

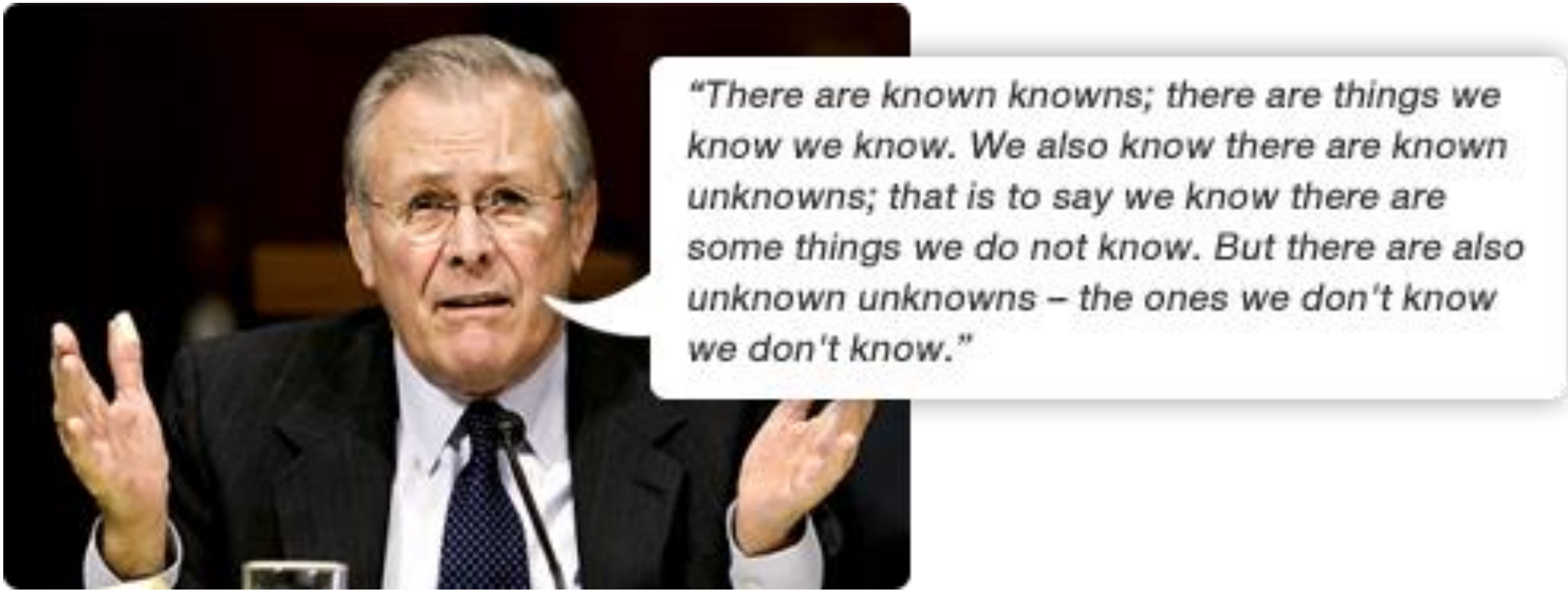
Errors

Where can you make errors?

The answer requires creativity



Errors

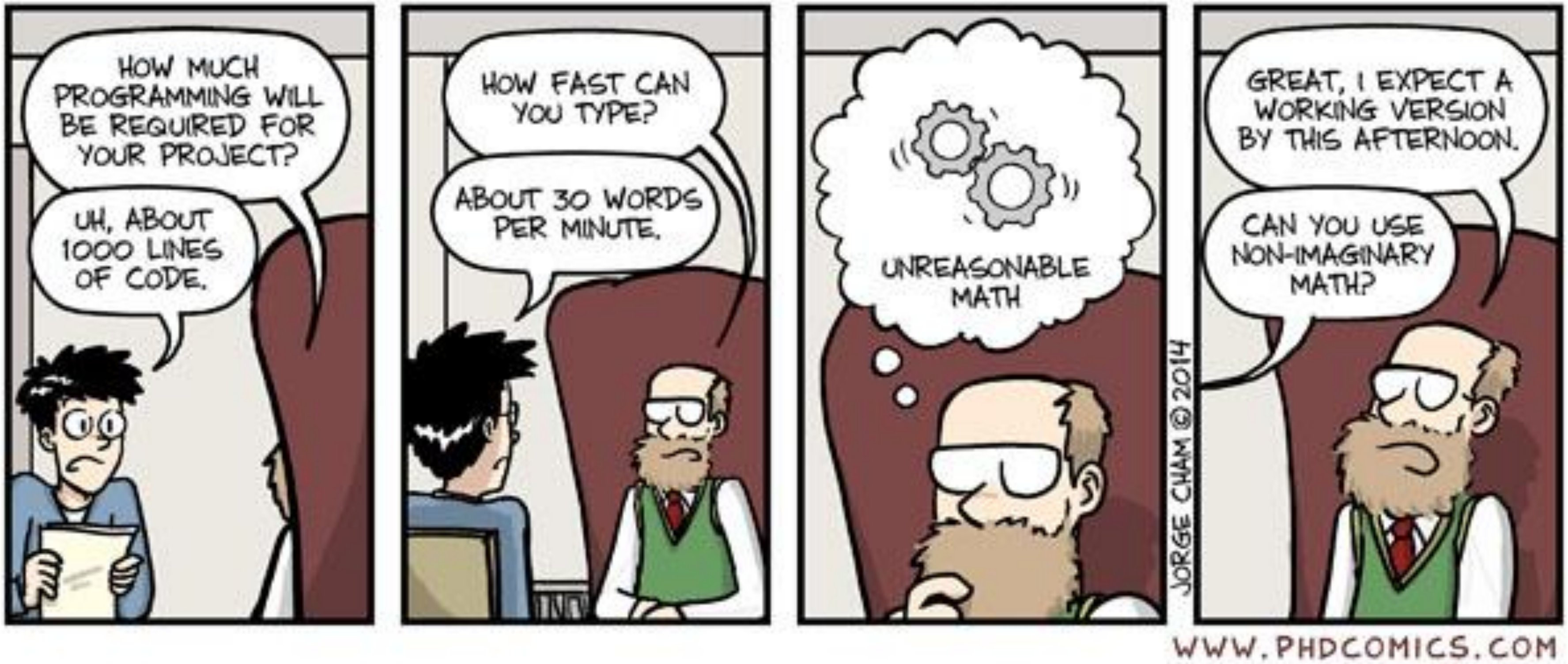


Donald Rumsfeld, then U.S. Secretary of Defense, www.fundraisingcollective.com

How can you find errors?

Ask questions!

1. Did I use the correct theory?
2. Are all relevant phenomena taken into consideration?
3. Did I make an error in calculation?
4. Did I make an error in my computer program?
5. What is the effect of discretization? (e.g. mesh size, taking numerical derivatives, numerical integration)
6. What is the effect of my assumptions? (e.g. linear behavior)
7. How reliable is my input data?
8. What is the accuracy of the model?
9. Is the model validated?


HOW MUCH PROGRAMMING WILL BE REQUIRED FOR YOUR PROJECT?
UH, ABOUT 1000 LINES OF CODE.
HOW FAST CAN YOU TYPE?
ABOUT 30 WORDS PER MINUTE.
UNREASONABLE MATH
JORGE CHAM © 2014
GREAT, I EXPECT A WORKING VERSION BY THIS AFTERNOON.
CAN YOU USE NON-IMAGINARY MATH?
WWW.PHDCOMICS.COM

Error in computer program?

```
      KK=K4+KSPAN
      IF (KK.LT.NN) GOTO 520
      KK=KK-NN
      IF (KK.LE.KSPAN) GOTO 520
      GOTO 700
C TRANSFORM FOR ODD FACTORS
600   K=NFAC(I)
      KSPNN=KSPAN
      KSPAN=KSPAN/K
      IF (K.EQ.3) GOTO 320
      IF (K.EQ.5) GOTO 510
      IF (K.EQ.JF) GOTO 640
      JF=K
      S1=RAD/DBLE(K)
      C1=DCOS(S1)
      S1=DSIN(S1)
      IF (JF.GT.MAXF) GOTO 998
      CK(JF)=1D0
      SK(JF)=0D0
      J=1
630   CK(J)=CK(K)*C1+SK(K)*S1
      SK(J)=CK(K)*S1-SK(K)*C1
      K=K-1
      CK(K)=CK(J)
      SK(K)=-SK(J)
      J=J+1
      IF (J.LT.K) GOTO 630
640   K1=KK
      K2=KK+KSPNN
      AA=A(KK)
      BB=B(KK)
      AK=AA
      BK=BB
      J=1
```

Industry Average: ″about 15 – 50 errors per 1000 lines of delivered code.″

Space Shuttle: "0 errors in 500,000 lines of code″

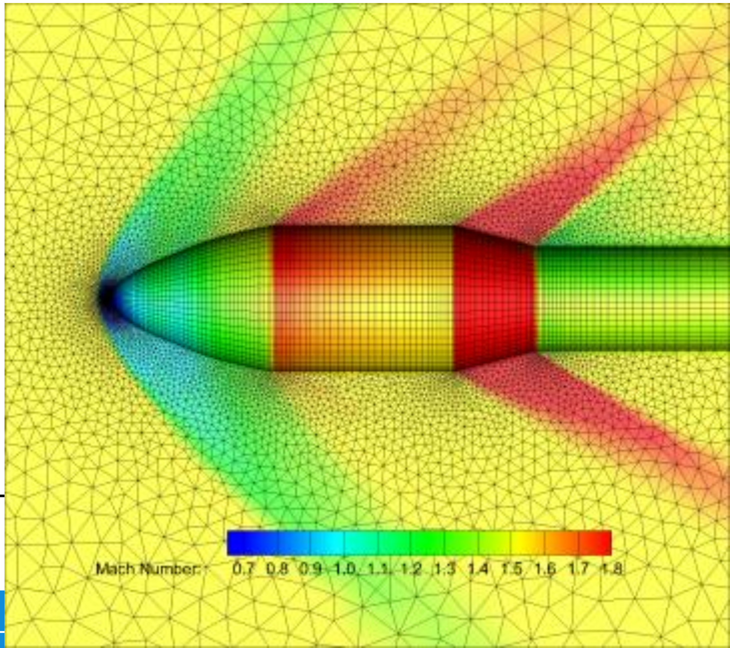
Source: Code Complete, Steve McConnell

Compare Numerical model to another model

- Numerical model: discretization in space (finite-elements) and time (numerical integration).

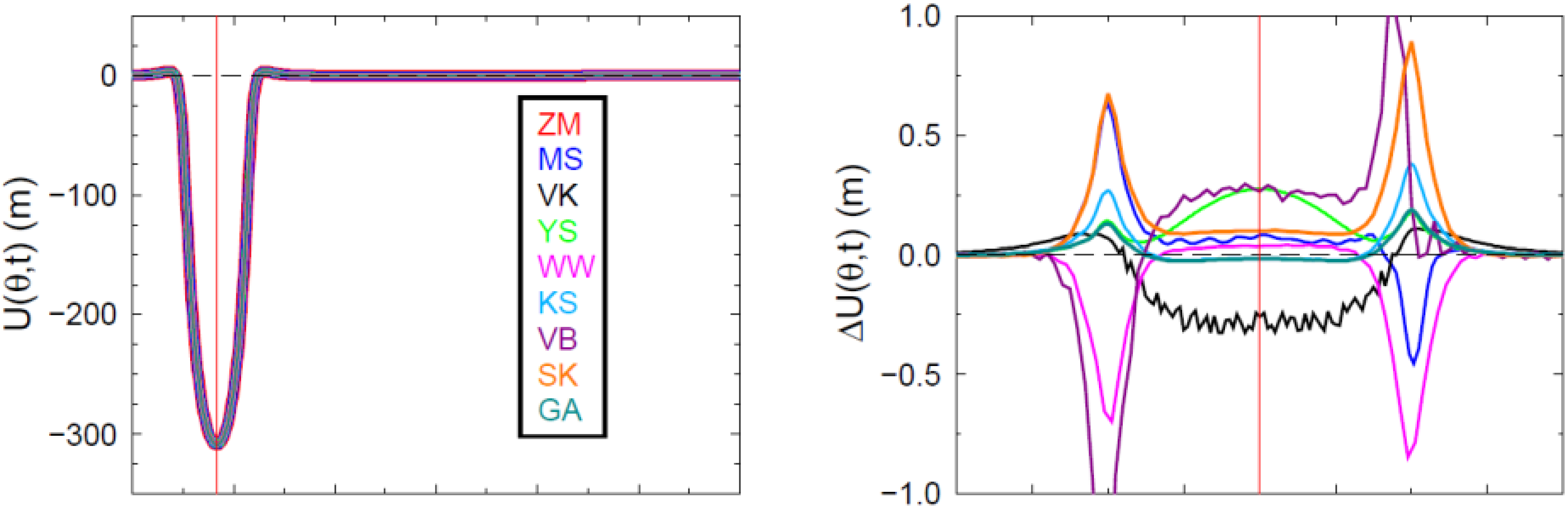
- Simpler (Analytical) model: check for the implementation of the numerical model, test discretization.

Question: When comparing the numerical model to an analytical model do you want to take the same assumptions as in the analytical model?



Error in computer program?

Check against an independent model



Martinec, Klemann, van der Wal, et al. (2018)

How can you find errors?

Ask the right questions!

1. Did I use the correct theory?
2. Are all relevant phenomena taken into consideration?
3. Did I make an error in calculation?
4. Did I make an error in my computer program?
5. What is the effect of discretization? (e.g. mesh size, taking numerical derivatives, numerical integration)
6. What is the effect of my assumptions? (e.g. linear behavior)
7. How reliable is my input data?
8. What is the accuracy of the model?
9. Is the model validated?

How can you find errors?

Ask the right questions!

1. Did I use the correct theory?
2. Are all relevant phenomena taken into consideration?
3. Did I make an error in calculation?
4. Did I make an error in my computer program?
5. What is the effect of discretization? (e.g. mesh size, taking numerical derivatives, numerical integration)
6. What is the effect of my assumptions? (e.g. linear behavior)
7. How reliable is my input data?
8. What is the accuracy of the model?
9. Is the model validated?

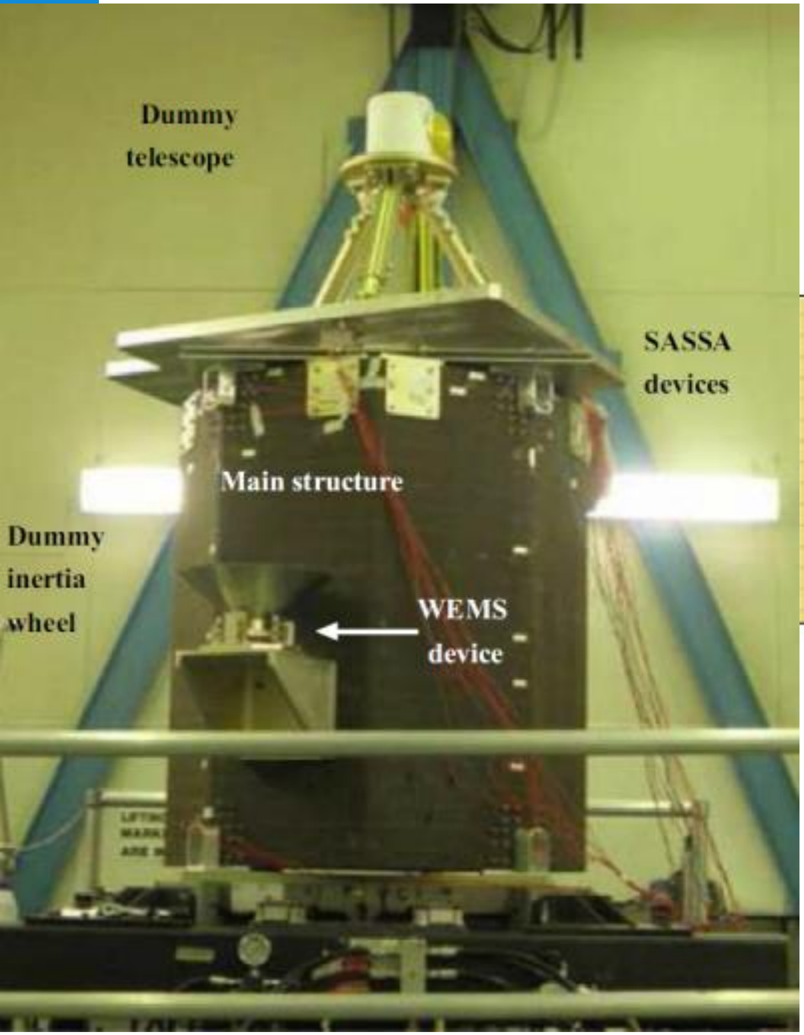
Validation

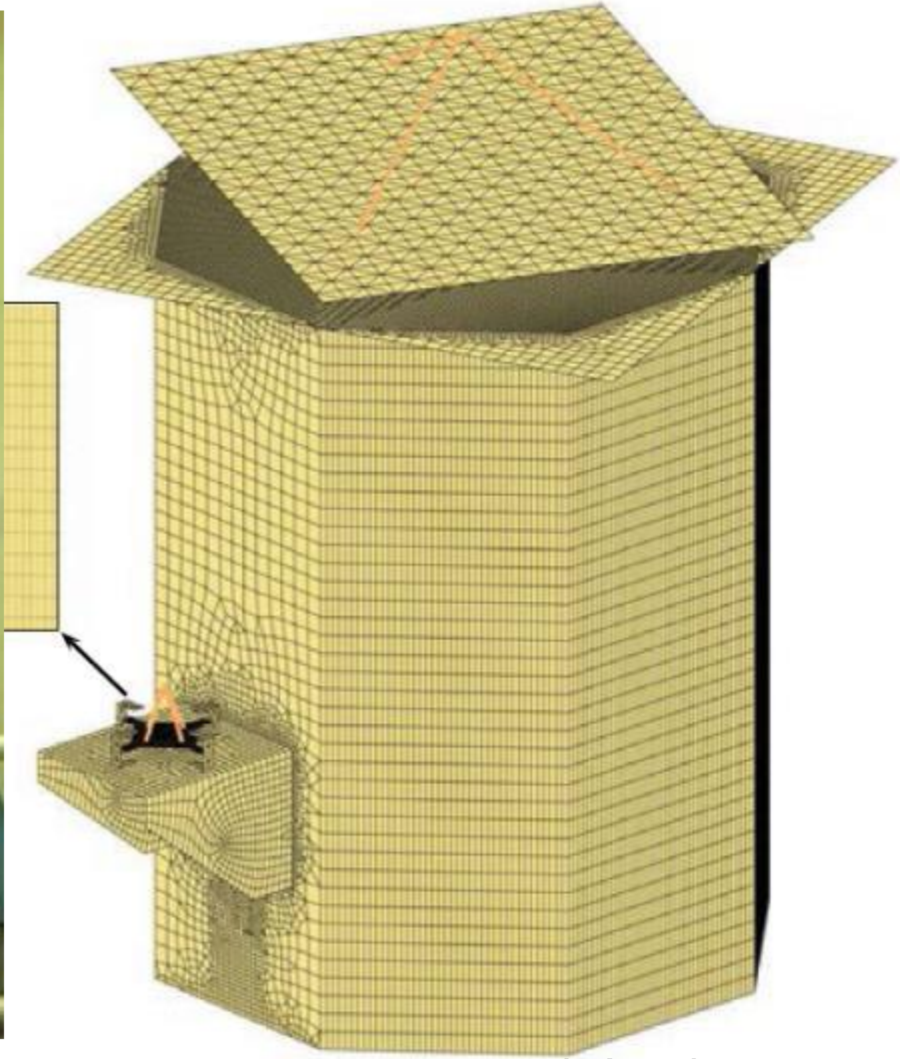
“Determine if the simulation results accurately represent the physical problem”

Confrontation with reality

Validation

SmallSat structure, EADS-Astrium





Renson et al, (Nonlinear Dynamics, 2015)

Validation

Mode #	Model freq. (Hz)	Experimental freq. (Hz)
1	8.06	8.19
2	9.14	–
3	20.44	–
4	21.59	–
5	22.05	20.18

Renson et al., (Nonlinear Dynamics, 2015)

How can you find errors?

Ask the right questions!

1. Did I use the correct theory?
2. Are all relevant phenomena taken into consideration?
3. Did I make an error in calculation?
4. Did I make an error in my computer program?
5. What is the effect of discretization? (e.g. mesh size, taking numerical derivatives, numerical integration)
6. What is the effect of my assumptions? (e.g. linear behavior)
7. How reliable is my input data?
8. What is the accuracy of the model?
9. Is the model validated?

How can you find errors?

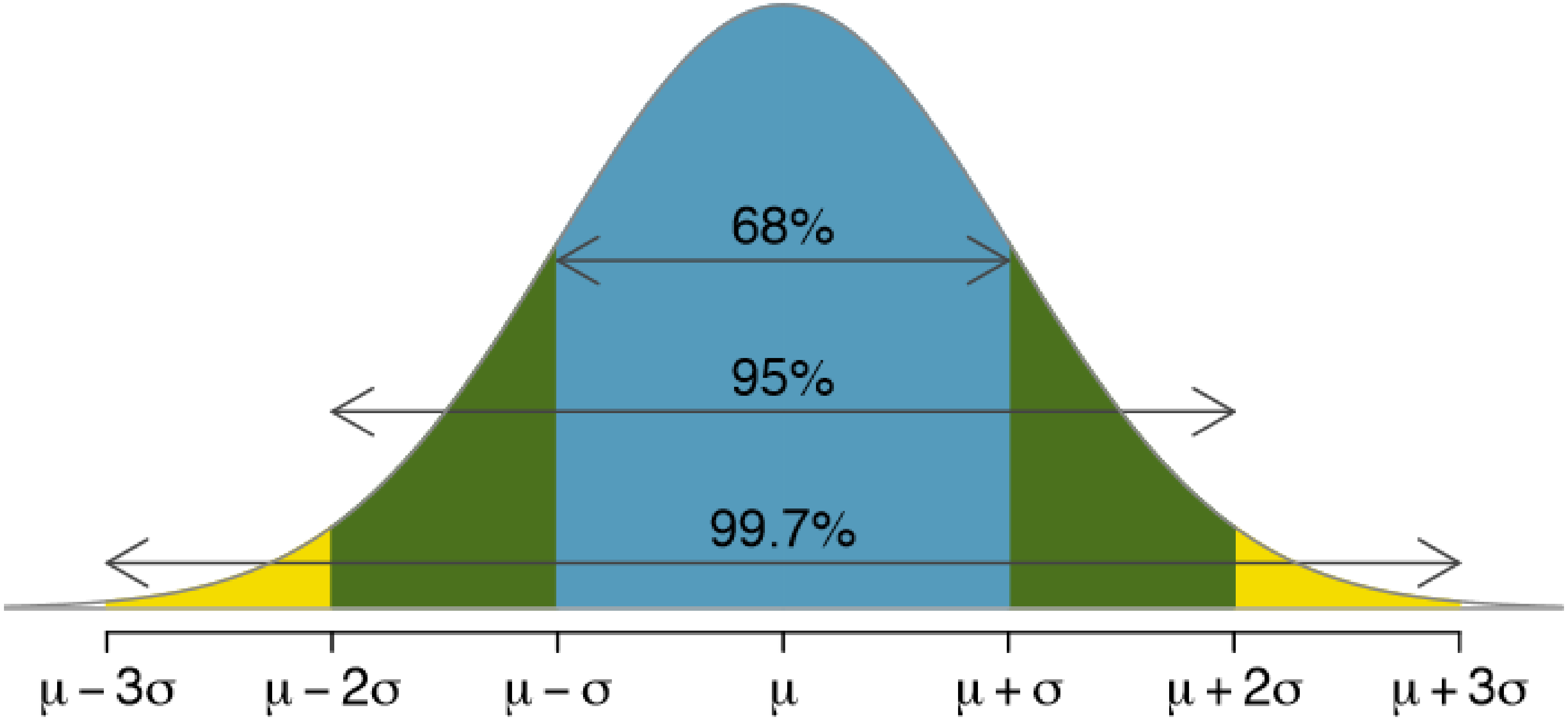
Ask the right questions!

1. Did I use the correct theory?
2. Are all relevant phenomena taken into consideration?
3. Did I make an error in calculation?
4. Did I make an error in my computer program?
5. What is the effect of discretization? (e.g. mesh size, taking numerical derivatives, numerical integration)
6. What is the effect of my assumptions? (e.g. linear behavior)
7. How reliable is my input data?
8. What is the accuracy of the model?
9. Is the model validated?

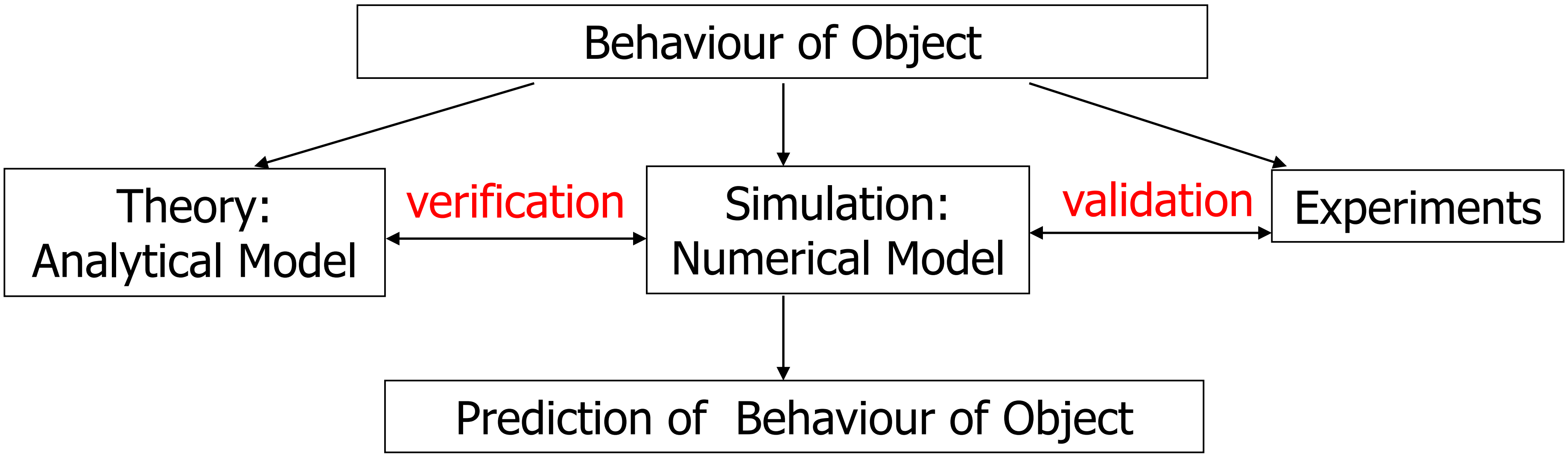
Evaluation

Example of hypothesis: The simulation predicts the correct natural frequency within 10% in 95% of the cases.

Hypothesis testing



Summary: Verification and Validation



Quiz: Verification or Validation

1) On an exam you have to derive an equation. You check if the equation has the right units
2) You write a computer program for your MSc thesis and check that you get the same result as the MSc student before you.
3) A self-driving car prototype driving on the road with a test driver on stand-by
4) You have built a simulation for the ocean and check that the difference between high and low tide is 12 hours and 25 minutes

Verification & Validation

V & V might not be the most glamorous, but..

- V &V procedures are required for accreditation of a model or simulation
- V & V procedures are required for certification of aerospace vehicles with operational requirements

AIAA Guide for the Verification and Validation of Computational Fluid Dynamics Simulations

Why Verify? (Hardware)

Mistakes cost money:

- Defective Product

  Pentium FDIV bug cost Intel US$475 million.

- Engineering Change Orders

  Each step of design cycle, bugs cost 10x more.

- Late to Market

  6 mo late on 5 yr lifespan loses 30% of total profit.

Alan Hu, Computer Science, UBC

Verification and Validation in the BSc/MSc

- V & V procedures are to be defined in the midterm phase of the Design Synthesis Exercise
- V & V procedures are to be executed for the final design in the DSE
- V & V procedures will be of great help during your MSc thesis work!

SOFTWARE VERIFICATION

Verification of Software

When somebody gives you a program, can you check if it is functioning properly?

No, you need a *specification* of the program

Language can be imprecise, mathematics is precise but not readable

Laski and Stanley (2009)

It is very difficult to be an objective tester of your own code. Working in a group has advantages!

Specification (what and how)

Pseudocode:

begin

Compute the set *S* of all monotonically increasing sequences in the subarray *A* [1 ... *n*];

Find the longest length of the sequences in S ;

end ;

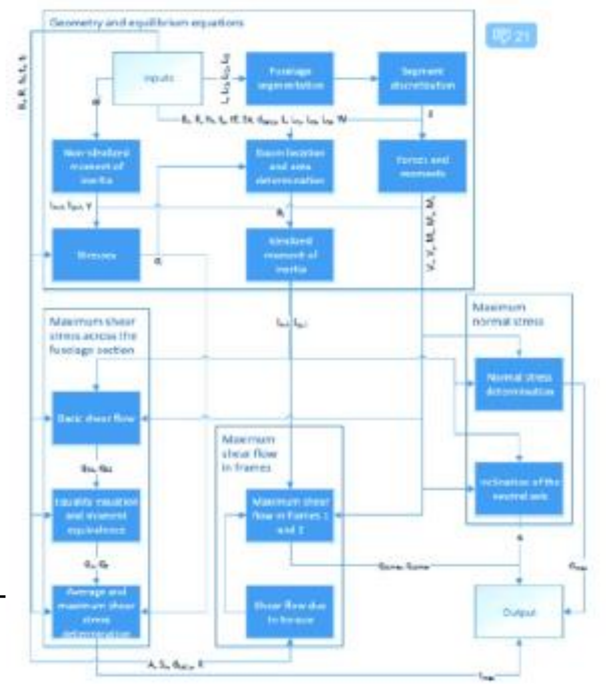


Figure 3.1: Flow chart

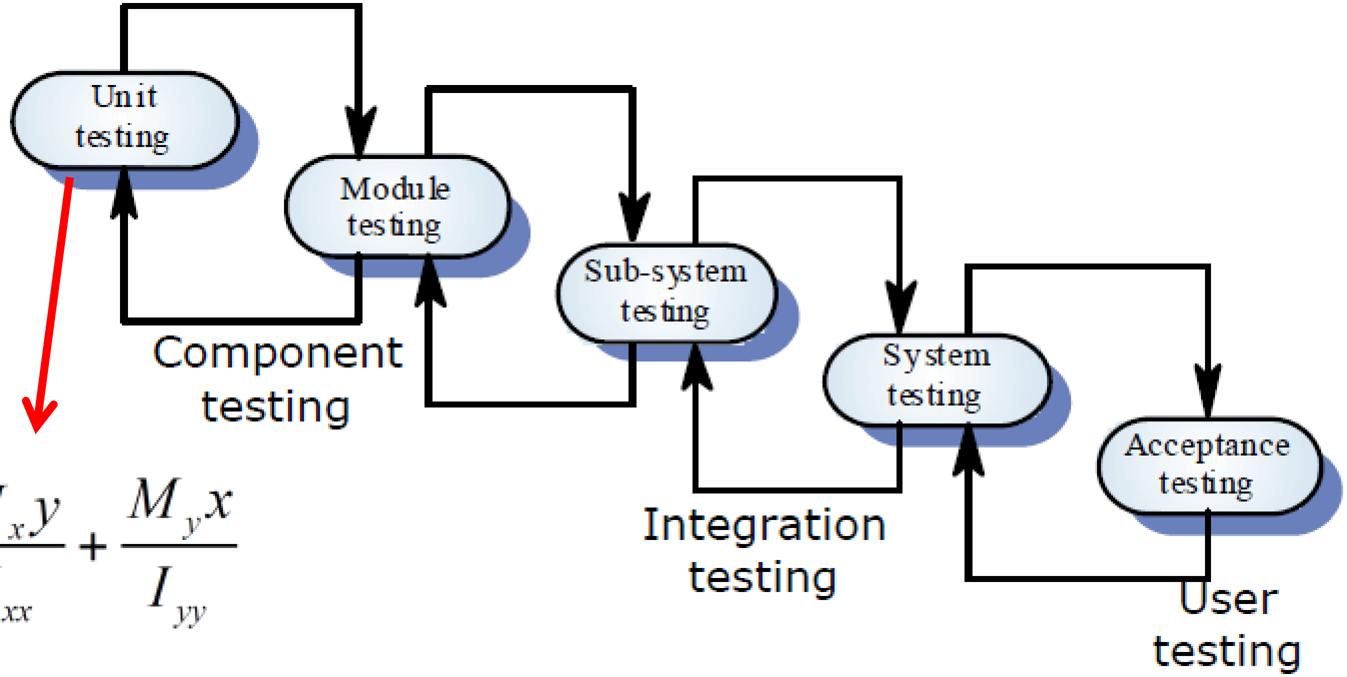
$$q_{s,f} = -\frac{I_{xx}V_x - I_{xy}V_y}{I_{xx}I_{yy} - I_{xy}^2}\sum_{r=1}^{n} B_r y_r - \frac{I_{yy}V_y - I_{xy}V_x}{I_{xx}I_{yy} - I_{xy}^2}\sum_{r=1}^{n} B_r x_r + q_{s0,f}$$

19

$$q_{s0f} = \frac{V_x\eta - V_y\varepsilon - \int_0^{\theta} r^2 q_{bf}\, d\theta}{2A}$$

Laski and Stanley (2009)

Software development



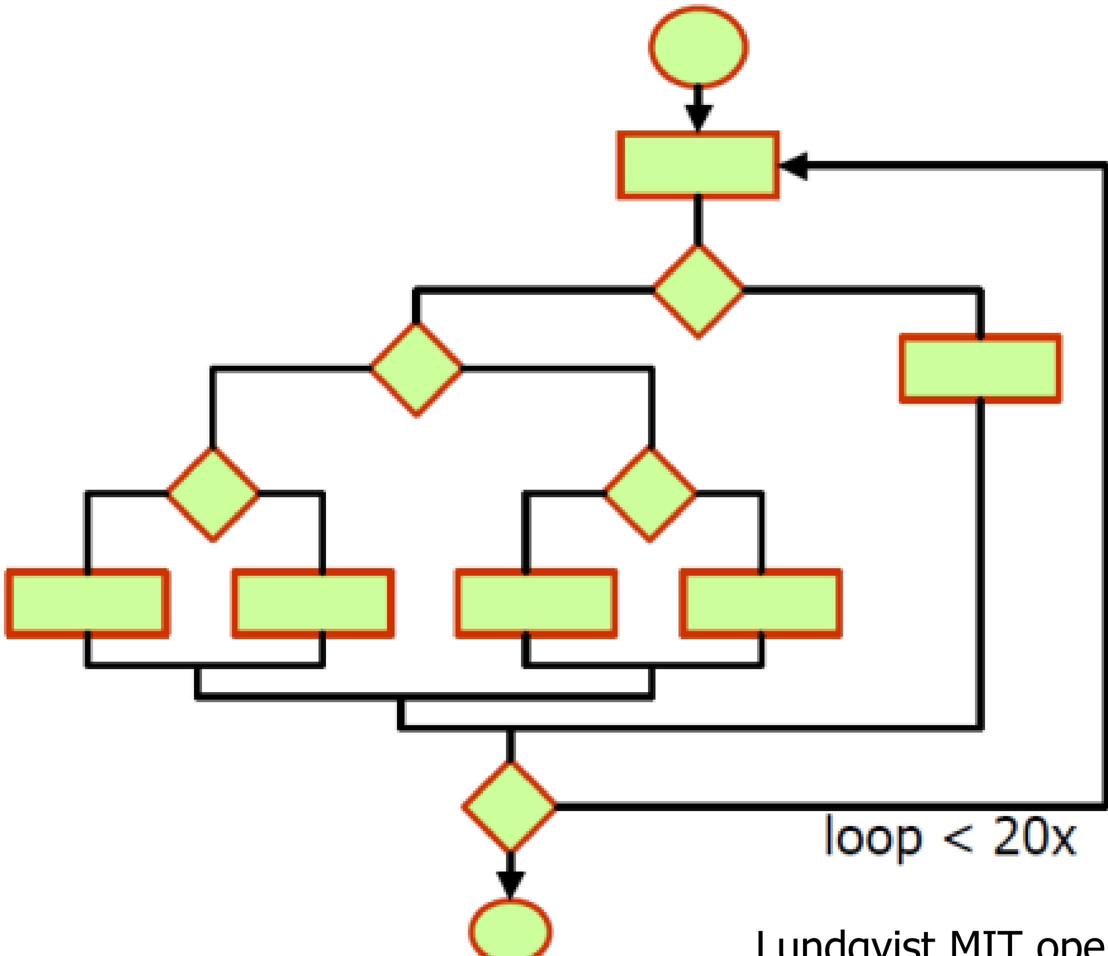
$$\sigma_z = \frac{M_x y}{I_{xx}} + \frac{M_y x}{I_{yy}}$$

Verification does not consist of only one comparison!

Lundqvist MIT open CourseWare

Verification of Software

Testing can only show absence of *incorrectness*



There are $5^{20}=10^{14}$ possible paths

Lundqvist MIT open CourseWare

Testing (dynamic)

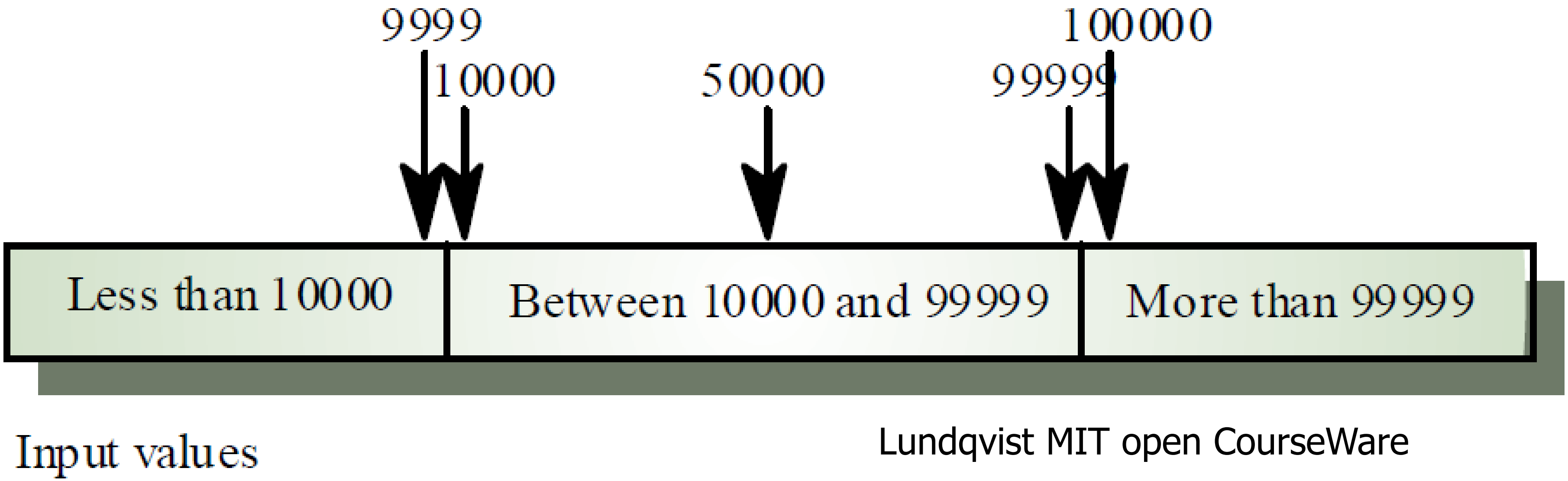
Intuition, common sense

Program to find the largest increasing sequence in an array A

t1 = (5, [5, 4 ,3, 2, 1])	(all sequences have length equal to one)
t2 = (5, [1, 2, 3, 4, 5])	(strictly increasing lengths of sequences)
t3 = (9, [1, 3, 5, 7, 9, 6, 4, 2, 0])	(increasing segment followed by a decreasing one)

Laski and Stanley (2009)

Testing (dynamic)



Lundqvist MIT open CourseWare

Structural testing (white box)

Code coverage: certain parts of the code are executed, provides a measure of how thorough the test is

```
Test        Percentage of                Percent
            Instruction Coverage         Branch (
------------------------------------------------
t1               87.7%                     66.(
t2               92.3%                     75.(
------------------------------------------------
Not executed:
Instructions: 14, 15
Branches:        (13 14), (14 16), (14 15)
```

```
Function monotone(
         A : Int_Array; { array[1..20] of integer }
         n : integer )  { size of the defined lower }
           : integer ;  { portion of A }
VAR
  i , {index for current limseq }
  j , {index for predecessors of current limseq }
  maxj,     {length of current longest predecessor subsequence}
  pmax,              { end of current limseq in A[1..i-1] }
  curr,              { = A[i] }
  maxl : integer;    { length of limseq ending at pmax }
  length: Int_Array; { length[k] is the length of}
                     { limseq at k }
    begin   { monotone }
{ 1}   { <STAD> Initialization of parameter A }
{ 2}   { <STAD> Initialization of parameter n }
{ 3}   length[ 1 ] := 1 ;
{ 4}   pmax := 1 ;
{ 5}   maxl := 1 ;
{ 6}   i := 2 ;
{ 7}   while i <= n do
         begin
{ 8}       curr := A[ i ] ;
{ 9}       if curr < A[ pmax ] then
             begin
{10}           max j:= 1 ;
{11}           j := 1 ;
{12}           while j <=( i - 1 ) do
                 begin
{13}               if A[ j ] < curr then
                     begin
{14}                   if maxj < length[ j ] then
{15}                     maxj := length[ j ] ;
                     end ;
{16}               j := j + 1 ;
                 end ;
{17}           length[ i ] := maxj + 1 ;
{18}           if length[ i ] > maxl then
                 begin
{19}               maxl := maxl + 1 ;
{20}               pmax := i ;
                 end ;
             end
           ELSE    { if curr < A[ pmax ] }
             begin
{ 21}          maxl := maxl + 1 ;
{ 22}          length[ i ] := maxl ;
{ 23}          pmax := i ;
             end ;
```

Debugging

- “walking through” the code vs intuition

- Binary search

After the fix, retest!

> *The sad truth is that debugging is the least researched and, consequently, least understood area in software engineering, despite the fact that it is most likely one of the most time-consuming and costly activities.* Laski and Stanley (2009)

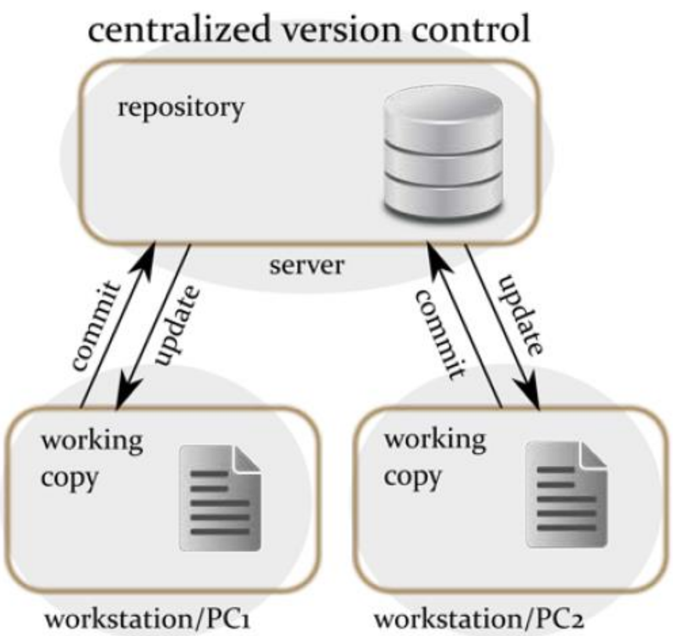
Tools – AE1205 lecture notes

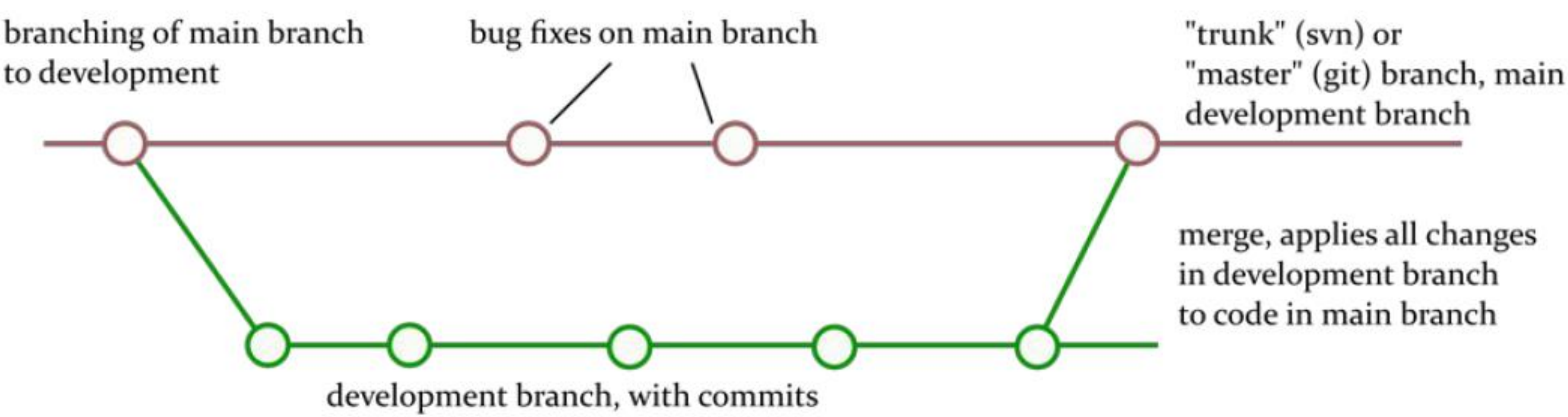
4. Making your code reusable and readable

File Edit Format Run Options Windows Help

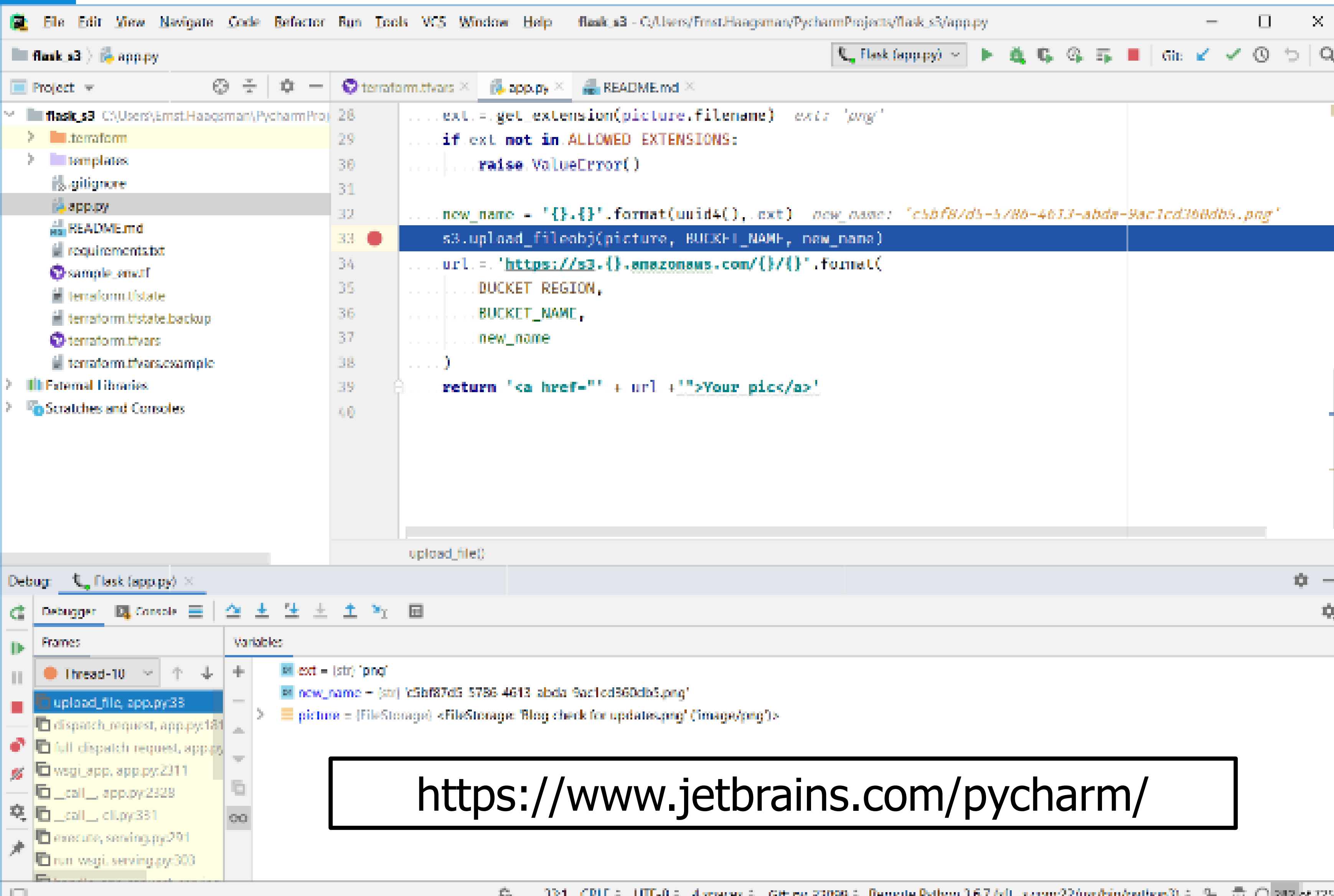
```
def solveabc(a,b,c):

# Function solveabc solves quadratic equation:
#                 a x2 +  b x  + c = 0 for x
#
# Input: a,b,c = coefficients of polynomials (floats or integer)
# Output: list with 0,1 or 2 solutions for x (floats)
#
# User should check number of solutions by
# checking length of list returned as result

# Calculate discriminant
    D = b*b-4*a*c
```

Tools – AE1205 Version Control





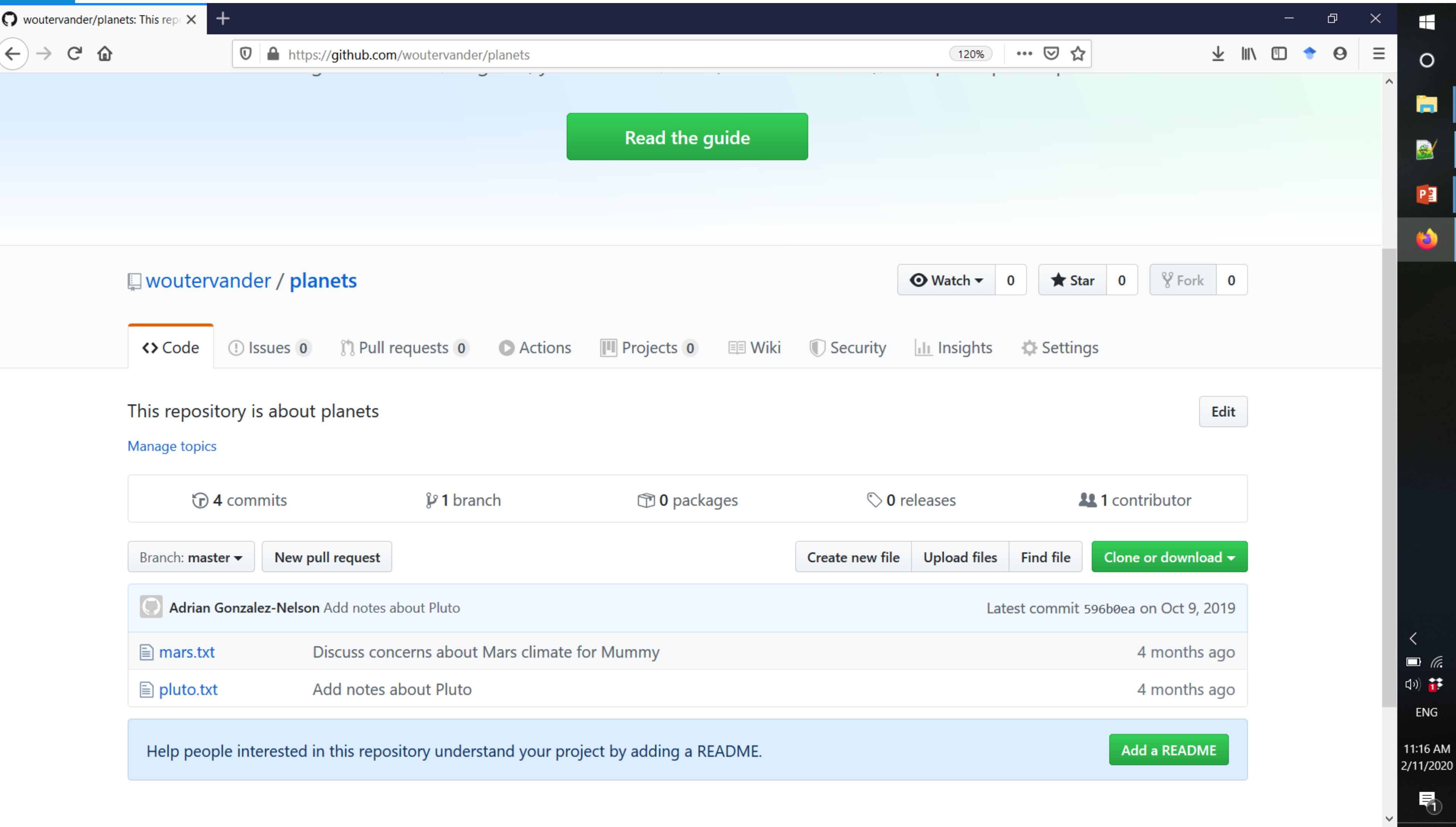
Debugging tools - Pycharm



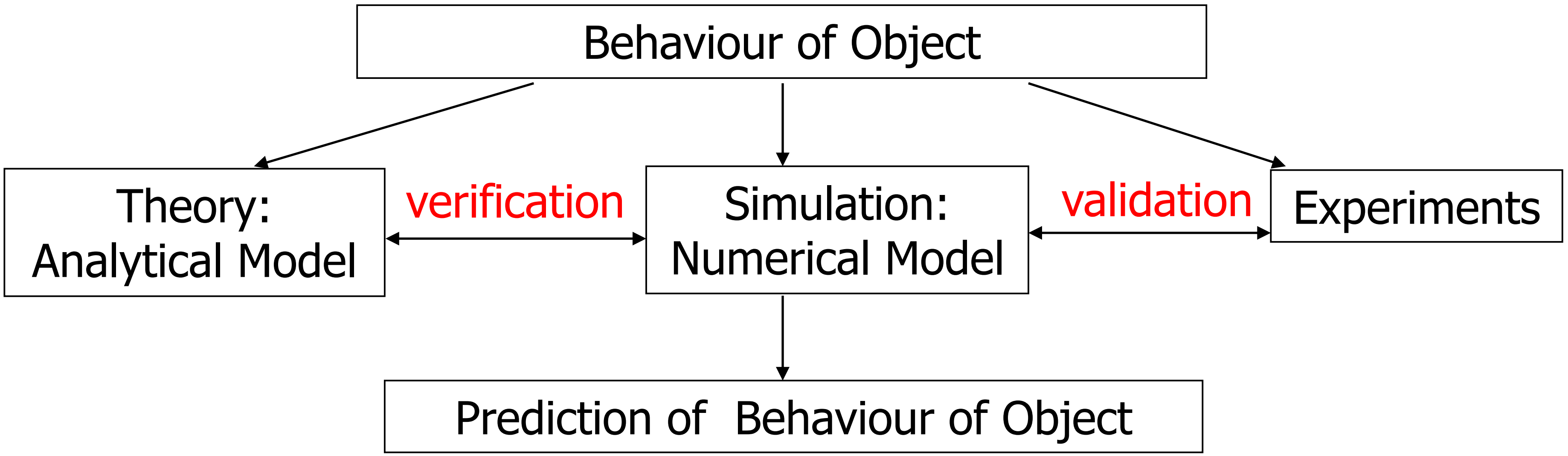
https://www.jetbrains.com/pycharm/

Debugging tools - Github

https://github.com



Summary: Verification and Validation



Verification: “Are you solving it right?”
Validation: “Are you solving the right thing?”