Software engineering during the Italian Renaissance

And some other stuff. It will not be a rant, I promise.

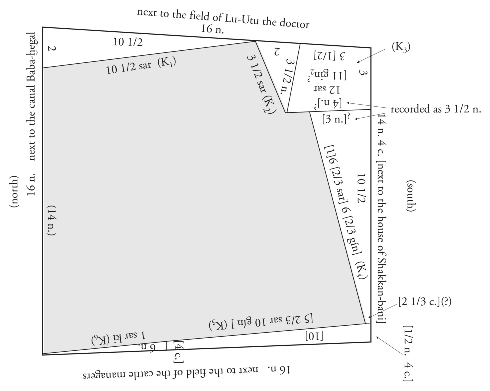
Michele Finelli
m@biodec.com
BioDec

Index

Index

Once upon a time in Babylonia . . .

. . . they had problems like measuring the size of fields, after a flood. As a consequence, they developed geometry (in Egypt too).



“A Geometric Algorithm with Solutions to Quadratic Equations in a Sumerian Juridical Document from Ur III Umma” by Jöran Friberg, Cuneiform Digital Library Journal 2009:3, ISSN 1540-8779.

The quadratic equation

- To solve their problems they asked for solution to equations, in particular to instances of the **quadratic equation**.
- You know, the stuff you studied in high school.
- Like that:

$$ax^2 + bx + c = 0$$

with $a \neq 0$.
- The curve $y = ax^2 + bx + c$ is a **parabola** with the axis of symmetry parallel to the ordinate axis.

The quadratic equation

- To solve their problems they asked for solution to equations, in particular to instances of the **quadratic equation.**
- You know, the stuff you studied in high school.
- Like that:

$$ax^2 + bx + c = 0$$

with $a \neq 0$.
- The curve $y = ax^2 + bx + c$ is a **parabola** with the axis of symmetry parallel to the ordinate axis.

BIODEC
Evolving ICT Infrastructures

The quadratic equation

- To solve their problems they asked for solution to equations, in particular to instances of the **quadratic equation**.
- You know, the stuff you studied in high school.
- Like that:

$$ax^2 + bx + c = 0$$

with $a \neq 0$.
- The curve $y = ax^2 + bx + c$ is a **parabola** with the axis of symmetry parallel to the ordinate axis.

The quadratic equation

- To solve their problems they asked for solution to equations, in particular to instances of the **quadratic equation.**
- You know, the stuff you studied in high school.
- Like that:

$$ax^2 + bx + c = 0$$

with $a \neq 0$.

- The curve $y = ax^2 + bx + c$ is a **parabola** with the axis of symmetry parallel to the ordinate axis.

BIODEC
Evolving ICT Infrastructures

The quadratic equation

- To solve their problems they asked for solution to equations, in particular to instances of the **quadratic equation**.
- You know, the stuff you studied in high school.
- Like that:

$$ax^2 + bx + c = 0$$

  with $a \neq 0$.
- The curve $y = ax^2 + bx + c$ is a **parabola** with the axis of symmetry parallel to the ordinate axis.

A formula

- Just in case you forgot, the solution of the quadratic equation …
- …it is given by the following formula:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

A formula

- Just in case you forgot, the solution of the quadratic equation . . .
- . . . it is given by the following formula:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

A formula

- Just in case you forgot, the solution of the quadratic equation ...
- ...it is given by the following formula:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

But . . .

That formula is not what was used in Babylonia, since they had trouble acknowledging the existence of **negative numbers.**

Instead ...

They distinguished three **different** cases, where $b, c \geq 0$:

1. $x^2 + bx = c$
2. $x^2 + c = bx$
3. $x^2 = bx + c$

(notice also that they put $a = 1$ and divided the other coefficients accordingly).

So . . .

In a sense, they had **different algorithms** because of the different representations of the **same** equation.

Index

Fast forward few millennia

- It's year 1494, Luca Pacioli publishes "Summa de Arithmetica, Geometria, Proportioni e Proportionalità".
- At the time Pacioli is one of the most known and famous mathematicians of the world.
- He is also a designer.

Fast forward few millennia

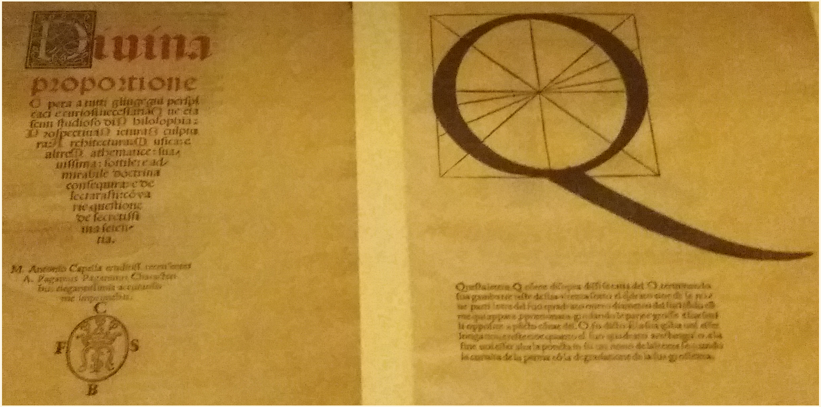
- It's year 1494, Luca Pacioli publishes “Summa de Arithmetica, Geometria, Proportioni e Proportionalità”.
- At the time Pacioli is one of the most known and famous mathematicians of the world.
- He is also a designer.

Fast forward few millennia

- It’s year 1494, Luca Pacioli publishes “Summa de Arithmetica, Geometria, Proportioni e Proportionalità”.
- At the time Pacioli is one of the most known and famous mathematicians of the world.
- He is also a designer.

Fast forward few millennia

- It's year 1494, Luca Pacioli publishes "Summa de Arithmetica, Geometria, Proportioni e Proportionalità".
- At the time Pacioli is one of the most known and famous mathematicians of the world.
- He is also a designer.


Diuina
proportione

Luca Pacioli

- But even great mathematicians make mistakes.
- In his *Summa de Arithmetica* he makes the conjecture that while there is a **general formula** to solve the quadratic equation, the **cubic equation** will probably never have one.

Luca Pacioli

- But even great mathematicians make mistakes.
- In his *Summa de Arithmetica* he makes the conjecture that while there is a **general formula** to solve the quadratic equation, the **cubic equation** will probably never have one.

Luca Pacioli

- But even great mathematicians make mistakes.
- In his *Summa de Arithmetica* he makes the conjecture that while there is a **general formula** to solve the quadratic equation, the **cubic equation** will probably never have one.

The cubic equation

- The cubic equation is written

$$ax^3 + bx^2 + cx + d = 0$$

- The $y = ax^3 + bx^2 + cx + d$ curve is the **cubic function**

The cubic equation

- The cubic equation is written

$$ax^3 + bx^2 + cx + d = 0$$

- The $y = ax^3 + bx^2 + cx + d$ curve is the **cubic function**

The cubic equation

- The cubic equation is written

$$ax^3 + bx^2 + cx + d = 0$$

- The $y = ax^3 + bx^2 + cx + d$ curve is the **cubic function**

The cubic function

This is the plot of a cubic function with three real roots:

Some Python code !

And since we are at a Plone conference, this is the Wikipedia Python code to plot the function:

```
# Author: Ika, 2013-07-24
from pylab import *
import pylab as pl
import numpy as np
pl.figure(figsize=(8,8), dpi=80)
# Create a plot of the cubic function y=(x+4)(x+1)(x-2)/4
x = np.linspace(-5.0,4.0,256, endpoint=True)
y = (x+4)*(x+1)*(x-2)/4
pl.plot(x,y,color="magenta", linewidth=3.0, linestyle="-")
# Set labels
pl.xlim(-5,4)
pl.ylim(-4,7)
# Move the spines
ax = pl.gca()
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.text(-5, 3, r'$y=\frac{1}{4}(x+4)(x+1)(x-2)$', fontsize=20)
# Set up the grid
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))
ax.grid(color="grey",linestyle='--',linewidth=1)
```

BIODEC
Evolving ICT Infrastructures

The cubic equation

- At the time it was also usual to make the substitution $x = x' - b/3a$ and then simplify it as $x'^3 + px' = q$.
- This amounts to shift the curve so that the flex point is on the ordinate axis.

The cubic equation

- At the time it was also usual to make the substitution $x = x' - b/3a$ and then simplify it as $x'^3 + px' = q$.
- This amounts to shift the curve so that the flex point is on the ordinate axis.

A mystery story

- Despite Luca Pacioli's conjecture . . .
- . . . in the year 1526, on his death bed, Scipione Dal Ferro tells Antonio Maria Del Fióre that he has found a formula to solve the cubic equation . . .

A mystery story

- Despite Luca Pacioli's conjecture ...
- ... in the year 1526, on his death bed, Scipione Dal Ferro tells Antonio Maria Del Fióre that he has found a formula to solve the cubic equation ...

Scipione Dal Ferro (or Scipione del Ferro)

- Scipione Dal Ferro (1465-1526) was a teacher at the university of Bologna, and he was defined "a matematico eccellentissimo".
- His wage increased six folds (25 lire in 1496, 150 lire after 1511) because of his mathematical skills.
- In fact it seems that he had devised a *secret software* to solve an equation in the form $x^3 + px = q \ldots$

Scipione Dal Ferro (or Scipione del Ferro)

- Scipione Dal Ferro (1465-1526) was a teacher at the university of Bologna, and he was defined "a matematico eccellentissimo".
- His wage increased six folds (25 lire in 1496, 150 lire after 1511) because of his mathematical skills.
- In fact it seems that he had devised a *secret software* to solve an equation in the form $x^3 + px = q \ldots$

Scipione Dal Ferro (or Scipione del Ferro)

- Scipione Dal Ferro (1465-1526) was a teacher at the university of Bologna, and he was defined "a matematico eccellentissimo".
- His wage increased six folds (25 lire in 1496, 150 lire after 1511) because of his mathematical skills.
- In fact it seems that he had devised a *secret software* to solve an equation in the form $x^3 + px = q \ldots$

Scipione Dal Ferro (or Scipione del Ferro)

- Scipione Dal Ferro (1465-1526) was a teacher at the university of Bologna, and he was defined "a matematico eccellentissimo".
- His wage increased six folds (25 lire in 1496, 150 lire after 1511) because of his mathematical skills.
- In fact it seems that he had devised a *secret software* to solve an equation in the form $x^3 + px = q \ldots$

Solving an equation is software, isn't it ?

There should be no surprise that finding the values of $x_{1,2}$ in

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

given the values of a, b and c, requires the creation and the execution of a (simple) algorithm.

A software with strange rules

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Where:

- **finding the specification is extremely difficult** (any small change to the formula means that you get **wrong** results);
- building the algorithm is **obvious** (once you know some basic mathematics);
- executing the algorithm can be worked in your head in a few seconds, if you train a little (*i.e.* **running the software is not the hard part**).

A software with strange rules

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Where:

- **finding the specification is extremely difficult** (any small change to the formula means that you get **wrong** results);
- building the algorithm is **obvious** (once you know some basic mathematics);
- executing the algorithm can be worked in your head in a few seconds, if you train a little (*i.e.* **running the software is not the hard part**).

A software with strange rules

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Where:

- **finding the specification is extremely difficult** (any small change to the formula means that you get **wrong** results);
- building the algorithm is **obvious** (once you know some basic mathematics);
- executing the algorithm can be worked in your head in a few seconds, if you train a little (*i.e.* **running the software is not the hard part**).

A software with strange rules

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Where:

- **finding the specification is extremely difficult** (any small change to the formula means that you get **wrong** results);
- building the algorithm is **obvious** (once you know some basic mathematics);
- executing the algorithm can be worked in your head in a few seconds, if you train a little (*i.e.* **running the software is not the hard part**).

Back to the sixteenth century

- Enter Gerolamo Cardano (1501-1576), Niccolò Tartaglia (1499-1557) and Lodovico Ferrari (1522-1565).
- Antonio Maria Del Fióre learned that Tartaglia was able to solve a cubic equation of the form $x^3 + x^2 = r$ and not believing him, challenged Tartaglia to a "disfida matematica" (a *mathematical fight*) in the year 1535.
- Tartaglia, preparing for the “battle”, discovered a **more general formula** to solve the cubic equation: in two hours he solved all Del Fióre challenges, while his opponent failed all Tartaglia’s.

Back to the sixteenth century

- Enter Gerolamo Cardano (1501-1576), Niccolò Tartaglia (1499-1557) and Lodovico Ferrari (1522-1565).
- Antonio Maria Del Fióre learned that Tartaglia was able to solve a cubic equation of the form $x^3 + x^2 = r$ and not believing him, challenged Tartaglia to a "disfida matematica" (a *mathematical fight*) in the year 1535.
- Tartaglia, preparing for the “battle”, discovered a **more general formula** to solve the cubic equation: in two hours he solved all Del Fióre challenges, while his opponent failed all Tartaglia’s.

Back to the sixteenth century

- Enter Gerolamo Cardano (1501-1576), Niccolò Tartaglia (1499-1557) and Lodovico Ferrari (1522-1565).
- Antonio Maria Del Fióre learned that Tartaglia was able to solve a cubic equation of the form $x^3 + x^2 = r$ and not believing him, challenged Tartaglia to a "disfida matematica" (a *mathematical fight*) in the year 1535.
- Tartaglia, preparing for the “battle”, discovered a **more general formula** to solve the cubic equation: in two hours he solved all Del Fióre challenges, while his opponent failed all Tartaglia’s.

Back to the sixteenth century

- Enter Gerolamo Cardano (1501-1576), Niccolò Tartaglia (1499-1557) and Lodovico Ferrari (1522-1565).
- Antonio Maria Del Fióre learned that Tartaglia was able to solve a cubic equation of the form $x^3 + x^2 = r$ and not believing him, challenged Tartaglia to a "disfida matematica" (a *mathematical fight*) in the year 1535.
- Tartaglia, preparing for the “battle”, discovered a **more general formula** to solve the cubic equation: in two hours he solved all Del Fióre challenges, while his opponent failed all Tartaglia’s.

Back to the sixteenth century

- Four years later (1539), Tartaglia revealed his discovery to Cardano, with an oath of secrecy.
- Cardano, in the following years, worked on finding a general solution on his own, together with his student Ferrari.
- In the year 1542 they stumbled on a Dal Ferro original paper, learning that he had found a solution years before, but never published it — Cardano felt that he was bound to keep the secret no more.
- When Cardano published his “Ars Magna” (1545) he gave the formulas for both **the solution to the cubic equation** and, together with Ferrari, **the solution to the quartic equation**.

Back to the sixteenth century

- Four years later (1539), Tartaglia revealed his discovery to Cardano, with an oath of secrecy.
- Cardano, in the following years, worked on finding a general solution on his own, together with his student Ferrari.
- In the year 1542 they stumbled on a Dal Ferro original paper, learning that he had found a solution years before, but never published it — Cardano felt that he was bound to keep the secret no more.
- When Cardano published his “Ars Magna” (1545) he gave the formulas for both **the solution to the cubic equation** and, together with Ferrari, **the solution to the quartic equation**.

Back to the sixteenth century

- Four years later (1539), Tartaglia revealed his discovery to Cardano, with an oath of secrecy.
- Cardano, in the following years, worked on finding a general solution on his own, together with his student Ferrari.
- In the year 1542 they stumbled on a Dal Ferro original paper, learning that he had found a solution years before, but never published it — Cardano felt that he was bound to keep the secret no more.
- When Cardano published his "Ars Magna" (1545) he gave the formulas for both **the solution to the cubic equation** and, together with Ferrari, **the solution to the quartic equation**.

Back to the sixteenth century

- Four years later (1539), Tartaglia revealed his discovery to Cardano, with an oath of secrecy.
- Cardano, in the following years, worked on finding a general solution on his own, together with his student Ferrari.
- In the year 1542 they stumbled on a Dal Ferro original paper, learning that he had found a solution years before, but never published it — Cardano felt that he was bound to keep the secret no more.
- When Cardano published his “Ars Magna” (1545) he gave the formulas for both **the solution to the cubic equation** and, together with Ferrari, **the solution to the quartic equation**.

Back to the sixteenth century

- Four years later (1539), Tartaglia revealed his discovery to Cardano, with an oath of secrecy.
- Cardano, in the following years, worked on finding a general solution on his own, together with his student Ferrari.
- In the year 1542 they stumbled on a Dal Ferro original paper, learning that he had found a solution years before, but never published it — Cardano felt that he was bound to keep the secret no more.
- When Cardano published his “Ars Magna” (1545) he gave the formulas for both **the solution to the cubic equation** and, together with Ferrari, **the solution to the quartic equation**.

Disfide

- Tartaglia published next year (1546) his “Quesiti et inventioni diverse”, where he claimed to have discovered the general solution of the cubic equation before Cardano (and, in fact, having revealed it to him).
- That started a long dispute over the paternity of the discovery, and Ferrari fought many *disfide* against Tartaglia, on behalf of his former mentor Cardano.
- Ferrari insisted on having oral expositions, while Tartaglia preferred written ones. The last *disfida* was lost by Tartaglia in Milan, the 10th of August of 1548, against Ferrari, mainly because of his stuttering problems (since the “Tartaglia” nickname).

Disfide

- Tartaglia published next year (1546) his “Quesiti et inventioni diverse”, where he claimed to have discovered the general solution of the cubic equation before Cardano (and, in fact, having revealed it to him).
- That started a long dispute over the paternity of the discovery, and Ferrari fought many *disfide* against Tartaglia, on behalf of his former mentor Cardano.
- Ferrari insisted on having oral expositions, while Tartaglia preferred written ones. The last *disfida* was lost by Tartaglia in Milan, the 10th of August of 1548, against Ferrari, mainly because of his stuttering problems (since the “Tartaglia” nickname).

Disfide

- Tartaglia published next year (1546) his “Quesiti et inventioni diverse”, where he claimed to have discovered the general solution of the cubic equation before Cardano (and, in fact, having revealed it to him).
- That started a long dispute over the paternity of the discovery, and Ferrari fought many *disfide* against Tartaglia, on behalf of his former mentor Cardano.
- Ferrari insisted on having oral expositions, while Tartaglia preferred written ones. The last *disfida* was lost by Tartaglia in Milan, the 10th of August of 1548, against Ferrari, mainly because of his stuttering problems (since the “Tartaglia” nickname).

Disfide

- Tartaglia published next year (1546) his “Quesiti et inventioni diverse”, where he claimed to have discovered the general solution of the cubic equation before Cardano (and, in fact, having revealed it to him).
- That started a long dispute over the paternity of the discovery, and Ferrari fought many *disfide* against Tartaglia, on behalf of his former mentor Cardano.
- Ferrari insisted on having oral expositions, while Tartaglia preferred written ones. The last *disfida* was lost by Tartaglia in Milan, the 10th of August of 1548, against Ferrari, mainly because of his stuttering problems (since the “Tartaglia” nickname).

And the winner is . . .

- The *disfide* are important historic documents: in one of the *carteggi* (papers exchanged by the contenders) Ferrari writes that he and Cardano read in Bologna a “libellum manu Scipionis Ferrei soceri sui iam diu conscriptum, in, quo istud inventum eleganter et docte explicatum tradebatur”.
- Moreover, Tartaglia himself was probably aware of Dal Ferro paternity (in a *carteggio* he wrote “questa particolarità non mi par licita a doverla disputare né manco negare”).
- At the time of the dispute Dal Ferro had been dead by twenty years.

And the winner is . . .

- The *disfide* are important historic documents: in one of the *carteggi* (papers exchanged by the contenders) Ferrari writes that he and Cardano read in Bologna a “libellum manu Scipionis Ferrei soceri sui iam diu conscriptum, in, quo istud inventum eleganter et docte explicatum tradebatur”.
- Moreover, Tartaglia himself was probably aware of Dal Ferro paternity (in a *carteggio* he wrote “questa particolarità non mi par licita a doverla disputare né manco negare”).
- At the time of the dispute Dal Ferro had been dead by twenty years.

And the winner is . . .

- The *disfide* are important historic documents: in one of the *carteggi* (papers exchanged by the contenders) Ferrari writes that he and Cardano read in Bologna a “libellum manu Scipionis Ferrei soceri sui iam diu conscriptum, in, quo istud inventum eleganter et docte explicatum tradebatur”.
- Moreover, Tartaglia himself was probably aware of Dal Ferro paternity (in a *carteggio* he wrote “questa particolarità non mi par licita a doverla disputare né manco negare”).
- At the time of the dispute Dal Ferro had been dead by twenty years.

And the winner is . . .

- The *disfide* are important historic documents: in one of the *carteggi* (papers exchanged by the contenders) Ferrari writes that he and Cardano read in Bologna a “libellum manu Scipionis Ferrei soceri sui iam diu conscriptum, in, quo istud inventum eleganter et docte explicatum tradebatur”.
- Moreover, Tartaglia himself was probably aware of Dal Ferro paternity (in a *carteggio* he wrote “questa particolarità non mi par licita a doverla disputare né manco negare”).
- At the time of the dispute Dal Ferro had been dead by twenty years.

The Dal Ferro-Tartaglia-Ferrari-Cardano formula

The formula for the cubic equation is the following:

$$x = \sqrt[3]{-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}$$

where the equation $ax^3 + bx^2 + cx + d = 0$ has been changed to the form

$$x^3 + px + q = 0$$

through the change $x \to x - b/3a$ (and a division by a).

BIODEC Evolving ICT Infrastructures

The Dal Ferro-Tartaglia-Ferrari-Cardano formula

In the formula you first compute the cube root:

$$u = \sqrt[3]{-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}$$

and then choose the value of the other cube root:

$$v = \sqrt[3]{-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}$$

for which the relation $uv = -p/3$ holds. Since a cubic root has three solutions over the complex field, you get three solutions for the original formula — the naïve application of the formula would end in *nine* values.

Strange phenomena

Take for example the equation $x^3 - 15x - 4 = 0$.

- Once you apply the formula you get

$$x = \sqrt[3]{2 + 11\sqrt{-1}} + \sqrt[3]{2 - 11\sqrt{-1}}$$

- Remember that at the time $\sqrt{-1}$ was literally garbage, a writing with no meaning at all, an obvious mistake.
- How could it be that the formula was right but you need other formulas to get the solutions $x_1 = 4$, $x_2 = \sqrt{3} - 2$ and $x_2 = -\sqrt{3} - 2$?

Strange phenomena

Take for example the equation $x^3 - 15x - 4 = 0$.

- Once you apply the formula you get

$$x = \sqrt[3]{2 + 11\sqrt{-1}} + \sqrt[3]{2 - 11\sqrt{-1}}$$

- Remember that at the time $\sqrt{-1}$ was literally garbage, a writing with no meaning at all, an obvious mistake.
- How could it be that the formula was right but you need other formulas to get the solutions $x_1 = 4$, $x_2 = \sqrt{3} - 2$ and $x_2 = -\sqrt{3} - 2$?

Strange phenomena

Take for example the equation $x^3 - 15x - 4 = 0$.

- Once you apply the formula you get

$$x = \sqrt[3]{2 + 11\sqrt{-1}} + \sqrt[3]{2 - 11\sqrt{-1}}$$

- Remember that at the time $\sqrt{-1}$ was literally garbage, a writing with no meaning at all, an obvious mistake.
- How could it be that the formula was right but you need other formulas to get the solutions $x_1 = 4$, $x_2 = \sqrt{3} - 2$ and $x_2 = -\sqrt{3} - 2$?

Strange phenomena

Take for example the equation $x^3 - 15x - 4 = 0$.

- Once you apply the formula you get

$$x = \sqrt[3]{2 + 11\sqrt{-1}} + \sqrt[3]{2 - 11\sqrt{-1}}$$

- Remember that at the time $\sqrt{-1}$ was literally garbage, a writing with no meaning at all, an obvious mistake.
- How could it be that the formula was right but you need other formulas to get the solutions $x_1 = 4$, $x_2 = \sqrt{3} - 2$ and $x_2 = -\sqrt{3} - 2$?

Strange phenomena

- Can we get rid of those strange $\sqrt{-1}$?
- Can we solve equations of any power order (*i.e.* quintic equations, and more) ?
- If not, can we prove the impossibility result ?

Strange phenomena

- Can we get rid of those strange $\sqrt{-1}$?
- Can we solve equations of any power order (*i.e.* quintic equations, and more) ?
- If not, can we prove the impossibility result ?

Strange phenomena

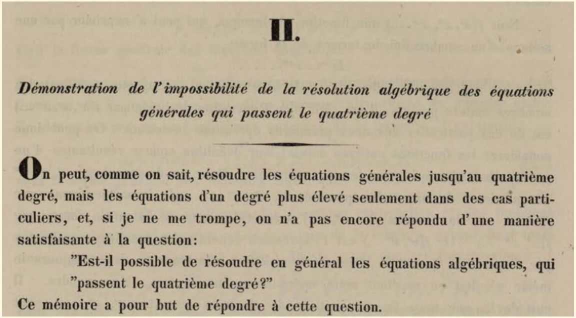
- Can we get rid of those strange $\sqrt{-1}$?
- Can we solve equations of any power order (*i.e.* quintic equations, and more) ?
- If not, can we prove the impossibility result ?

Strange phenomena

- Can we get rid of those strange $\sqrt{-1}$?
- Can we solve equations of any power order (*i.e.* quintic equations, and more) ?
- If not, can we prove the impossibility result ?

Quintic equation

These, and other questions were definitely settled **centuries** later: Lagrange (1770), the conjecture by Gauss (1798) and finally the first (incomplete) proof by Ruffini (1799) and then Abel (1824 and 1826).



II.

Démonstration de l'impossibilité de la résolution algébrique des équations générales qui passent le quatrième degré

On peut, comme on sait, résoudre les équations générales jusqu'au quatrième degré, mais les équations d'un degré plus élevé seulement dans des cas particuliers, et, si je ne me trompe, on n'a pas encore répondu d'une manière satisfaisante à la question:

"Est-il possible de résoudre en général les équations algébriques, qui "passent le quatrième degré?"

Ce mémoire a pour but de répondre à cette question.

Why did it take so long ?

In my opinion, because they lacked the **abstractions** needed to move forward.

- Babylonians did not acknowledge the existence of **negative numbers**.
- Dal Ferro and company were comfortable with irrational numbers (like $\sqrt{3}$) but did not know how to deal with **imaginary numbers**.
- Gauss and Abel were well acquainted with complex numbers, but it required **Galois theory**, by Èvariste Galois (1830), to develop a more general theory of algebraic equations.

Why did it take so long ?

In my opinion, because they lacked the **abstractions** needed to move forward.

- Babylonians did not acknowledge the existence of **negative numbers**.
- Dal Ferro and company were comfortable with irrational numbers (like $\sqrt{3}$) but did not know how to deal with **imaginary numbers**.
- Gauss and Abel were well acquainted with complex numbers, but it required **Galois theory**, by Èvariste Galois (1830), to develop a more general theory of algebraic equations.

Why did it take so long ?

In my opinion, because they lacked the **abstractions** needed to move forward.

- Babylonians did not acknowledge the existence of **negative numbers**.
- Dal Ferro and company were comfortable with irrational numbers (like $\sqrt{3}$) but did not know how to deal with **imaginary numbers**.
- Gauss and Abel were well acquainted with complex numbers, but it required **Galois theory**, by Èvariste Galois (1830), to develop a more general theory of algebraic equations.

Why did it take so long ?

In my opinion, because they lacked the **abstractions** needed to move forward.

- Babylonians did not acknowledge the existence of **negative numbers**.
- Dal Ferro and company were comfortable with irrational numbers (like $\sqrt{3}$) but did not know how to deal with **imaginary numbers**.
- Gauss and Abel were well acquainted with complex numbers, but it required **Galois theory**, by Èvariste Galois (1830), to develop a more general theory of algebraic equations.

Index

Similarities

- Is there a field where the lacking of abstractions is preventing progress ?
- I think it exists.
- It is called **software engineering**.

Similarities

- Is there a field where the lacking of abstractions is preventing progress ?
- I think it exists.
- It is called **software engineering.**

Similarities

- Is there a field where the lacking of abstractions is preventing progress ?
- I think it exists.
- It is called **software engineering.**

Similarities

- Is there a field where the lacking of abstractions is preventing progress ?
- I think it exists.
- It is called **software engineering**.

Questions

- How do we specify what a software should (and should not) do ?
- How do we validate the behaviour of the code we write ?
- How do we describe and communicate our architectures ?

Questions

- How do we specify what a software should (and should not) do ?
- How do we validate the behaviour of the code we write ?
- How do we describe and communicate our architectures ?

Questions

- How do we specify what a software should (and should not) do ?
- How do we validate the behaviour of the code we write ?
- How do we describe and communicate our architectures ?

Questions

- How do we specify what a software should (and should not) do ?
- How do we validate the behaviour of the code we write ?
- How do we describe and communicate our architectures ?

Answer

- In practice, the answer is more or less always **the same.**

Text

- Sentences, reports, tables, perhaps diagrams or pictures, devoid of any semantics except what a learned person could understand from actually reading the content.

BIODEC
Evolving ICT Infrastructures

Answer

- In practice, the answer is more or less always **the same.**

Text

- Sentences, reports, tables, perhaps diagrams or pictures, devoid of any semantics except what a learned person could understand from actually reading the content.

Answer

- In practice, the answer is more or less always **the same.**

Text

- Sentences, reports, tables, perhaps diagrams or pictures, devoid of any semantics except what a learned person could understand from actually reading the content.

Answer

- In practice, the answer is more or less always **the same.**

Text

- Sentences, reports, tables, perhaps diagrams or pictures, devoid of any semantics except what a learned person could understand from actually reading the content.

Don't get me wrong

- My position is not that we should resort to the kind of mathematics that was used to solve algebraic equations.
- The history of the discovery of their solutions was a tale to tell that in science progress usually stalls until we are able to go **from one language to another**.
- In the story I told, the new languages needed **new symbols** ($-, \sqrt{\cdot}, i$) and a **new understanding**.
- But, as a matter of fact, I see no new language (not a *new programming language*, of course) for software engineering.

Don’t get me wrong

- My position is not that we should resort to the kind of mathematics that was used to solve algebraic equations.
- The history of the discovery of their solutions was a tale to tell that in science progress usually stalls until we are able to go **from one language to another**.
- In the story I told, the new languages needed **new symbols** $(-, \sqrt{\cdot}, i)$ and a **new understanding**.
- But, as a matter of fact, I see no new language (not a *new programming language*, of course) for software engineering.

Don't get me wrong

- My position is not that we should resort to the kind of mathematics that was used to solve algebraic equations.
- The history of the discovery of their solutions was a tale to tell that in science progress usually stalls until we are able to go **from one language to another**.
- In the story I told, the new languages needed **new symbols** $(-, \sqrt{\cdot}, i)$ and a **new understanding**.
- But, as a matter of fact, I see no new language (not a *new programming language*, of course) for software engineering.

Don’t get me wrong

- My position is not that we should resort to the kind of mathematics that was used to solve algebraic equations.
- The history of the discovery of their solutions was a tale to tell that in science progress usually stalls until we are able to go **from one language to another**.
- In the story I told, the new languages needed **new symbols** ($-, \sqrt{\cdot}, i$) and a **new understanding**.
- But, as a matter of fact, I see no new language (not a *new programming language*, of course) for software engineering.

Don’t get me wrong

- My position is not that we should resort to the kind of mathematics that was used to solve algebraic equations.
- The history of the discovery of their solutions was a tale to tell that in science progress usually stalls until we are able to go **from one language to another**.
- In the story I told, the new languages needed **new symbols** ($-$, $\sqrt{\cdot}$, i) and a **new understanding**.
- But, as a matter of fact, I see no new language (not a *new programming language*, of course) for software engineering.

What we did instead

- I think that we, as the community of software practitioners, have focused too much on other issues.
- Take, for example the Agile Movement.

What we did instead

- I think that we, as the community of software practitioners, have focused too much on other issues.
- Take, for example the Agile Movement.

What we did instead

- I think that we, as the community of software practitioners, have focused too much on other issues.
- Take, for example the Agile Movement.

The agile movement

- The http://agilemanifesto.org/ (2001).
    1. *Individuals and Interactions* over processes and tools.
    2. *Working Software* over comprehensive documentation.
    3. *Customer Collaboration* over contract negotiation.
    4. *Responding to Change* over following a plan.
- It is a tale of improving the software development by tackling other issues (communication, organization, contracts, people).
- Agile achieved important results, but as software professionals we have the obligation to also improve our tools and techniques: **the practice of how we do the software thing**. But how ?

The agile movement

- The http://agilemanifesto.org/ (2001).
  1. *Individuals and Interactions* over processes and tools.
  2. *Working Software* over comprehensive documentation.
  3. *Customer Collaboration* over contract negotiation.
  4. *Responding to Change* over following a plan.
- It is a tale of improving the software development by tackling other issues (communication, organization, contracts, people).
- Agile achieved important results, but as software professionals we have the obligation to also improve our tools and techniques: **the practice of how we do the software thing**. But how ?

The agile movement

- The http://agilemanifesto.org/ (2001).
  1. *Individuals and Interactions* over processes and tools.
  2. *Working Software* over comprehensive documentation.
  3. *Customer Collaboration* over contract negotiation.
  4. *Responding to Change* over following a plan.
- It is a tale of improving the software development by tackling other issues (communication, organization, contracts, people).
- Agile achieved important results, but as software professionals we have the obligation to also improve our tools and techniques: **the practice of how we do the software thing**. But how ?

Unreasonably effective

COMMUNICATIONS ON PURE AND APPLIED MATHEMATICS, VOL. XIII, 001–14 (1960)

The Unreasonable Effectiveness of Mathematics in the Natural Sciences

Richard Courant Lecture in Mathematical Sciences delivered at New York University, May 11, 1959

EUGENE P. WIGNER
Princeton University

"and it is probable that there is some secret here which remains to be discovered." (*C. S. Peirce*)

There is a story about two friends, who were classmates in high school, talking about their jobs. One of them became a statistician and was working on population trends. He showed a reprint to his former classmate. The reprint started, as usual, with the Gaussian distribution and the statistician explained to his former classmate the meaning of the symbols for the actual population, for the average population, and so on. His classmate was a bit incredulous and was not quite sure whether the statistician was pulling his leg. "How can you know that?" was his query. "And what is this symbol here?" "Oh," said the statistician, "this is π." "What is that?" "The ratio of the circumference of the circle to its diameter." "Well, now you are pushing your joke too far," said the classmate, "surely the population has nothing to do with the circumference of the circle."

Not enough π ... so what ?

<insert the picture of some random venture capitalist from the Silicon Valley>

Yes, we do not have enough π and other mathematical stuff, but we did a pretty decent job: we built the Internet ! We disrupted and created whole markets ! The world IS different because of OUR software !

Be a better citizen of the world

- Yes, software built a **different** world.
- It should also be a **better** world.
- **Safety**, for example, is often left behind: our personal data are not safe, our privacy is not guaranteed, our digital rights are more often than not waived for a mistaken sense of security.
- It could be that software is building not a police state, but a **police world**.

Be a better citizen of the world

- Yes, software built a **different** world.
- It should also be a **better** world.
- **Safety**, for example, is often left behind: our personal data are not safe, our privacy is not guaranteed, our digital rights are more often than not waived for a mistaken sense of security.
- It could be that software is building not a police state, but a **police world**.

Be a better citizen of the world

- Yes, software built a **different** world.
- It should also be a **better** world.
- **Safety**, for example, is often left behind: our personal data are not safe, our privacy is not guaranteed, our digital rights are more often than not waived for a mistaken sense of security.
- It could be that software is building not a police state, but a **police world**.

Be a better citizen of the world

- Yes, software built a **different** world.
- It should also be a **better** world.
- **Safety**, for example, is often left behind: our personal data are not safe, our privacy is not guaranteed, our digital rights are more often than not waived for a mistaken sense of security.
- It could be that software is building not a police state, but a **police world**.

Be a better citizen of the world

- Yes, software built a **different** world.
- It should also be a **better** world.
- **Safety**, for example, is often left behind: our personal data are not safe, our privacy is not guaranteed, our digital rights are more often than not waived for a mistaken sense of security.
- It could be that software is building not a police state, but a **police world**.

Build better software (for the world)

- We need to **better understand** the systems we contribute to build.
- We need to be better in guaranteeing that these systems **will be safe to use**, that they will not be misused.
- We need to use better tools to design and build our software, to do **less mistakes**.
- We need to be better in **communicating the complexity** of what we build and its **implications in the real life** of our fellow citizens.

Build better software (for the world)

- We need to **better understand** the systems we contribute to build.
- We need to be better in guaranteeing that these systems **will be safe to use**, that they will not be misused.
- We need to use better tools to design and build our software, to do **less mistakes**.
- We need to be better in **communicating the complexity** of what we build and its **implications in the real life** of our fellow citizens.

Build better software (for the world)

- We need to **better understand** the systems we contribute to build.
- We need to be better in guaranteeing that these systems **will be safe to use**, that they will not be misused.
- We need to use better tools to design and build our software, to do **less mistakes**.
- We need to be better in **communicating the complexity** of what we build and its **implications in the real life** of our fellow citizens.

Build better software (for the world)

- We need to **better understand** the systems we contribute to build.
- We need to be better in guaranteeing that these systems **will be safe to use**, that they will not be misused.
- We need to use better tools to design and build our software, to do **less mistakes**.
- We need to be better in **communicating the complexity** of what we build and its **implications in the real life** of our fellow citizens.

Build better software (for the world)

- We need to **better understand** the systems we contribute to build.
- We need to be better in guaranteeing that these systems **will be safe to use**, that they will not be misused.
- We need to use better tools to design and build our software, to do **less mistakes**.
- We need to be better in **communicating the complexity** of what we build and its **implications in the real life** of our fellow citizens.

A quest

- We need a quest to find a **better theory of software**.
- "Nothing is more practical than a good theory." – Kurt Lewin.
- That theory may come from the only **unreasonably effective tool** that we have.

A quest

- We need a quest to find a **better theory of software.**
- “Nothing is more practical than a good theory.” – Kurt Lewin.
- That theory may come from the only **unreasonably effective tool** that we have.

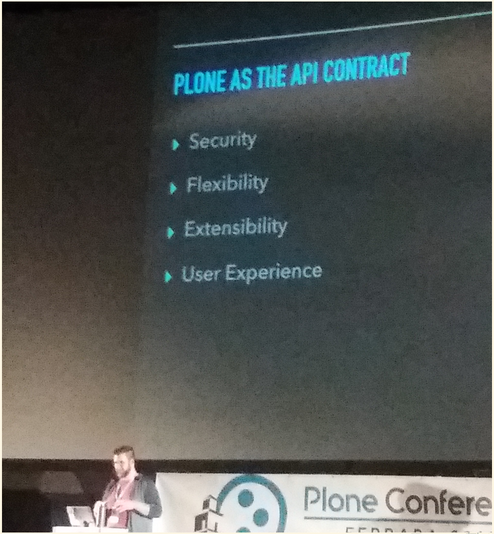
A quest

- We need a quest to find a **better theory of software.**
- “Nothing is more practical than a good theory.” – Kurt Lewin.
- That theory may come from the only **unreasonably effective tool** that we have.

A quest

- We need a quest to find a **better theory of software.**
- “Nothing is more practical than a good theory.” – Kurt Lewin.
- That theory may come from the only **unreasonably effective tool** that we have.

Yesterday !


PLONE AS THE API CONTRACT
Security
Flexibility
Extensibility
User Experience
Plone Confere

Thanks

Email m@biodec.com

Useless social @gaunilone

Incontro DevOps Italia See you in Bologna, March 5-6, 2020. https://2020.incontrodevops.it/