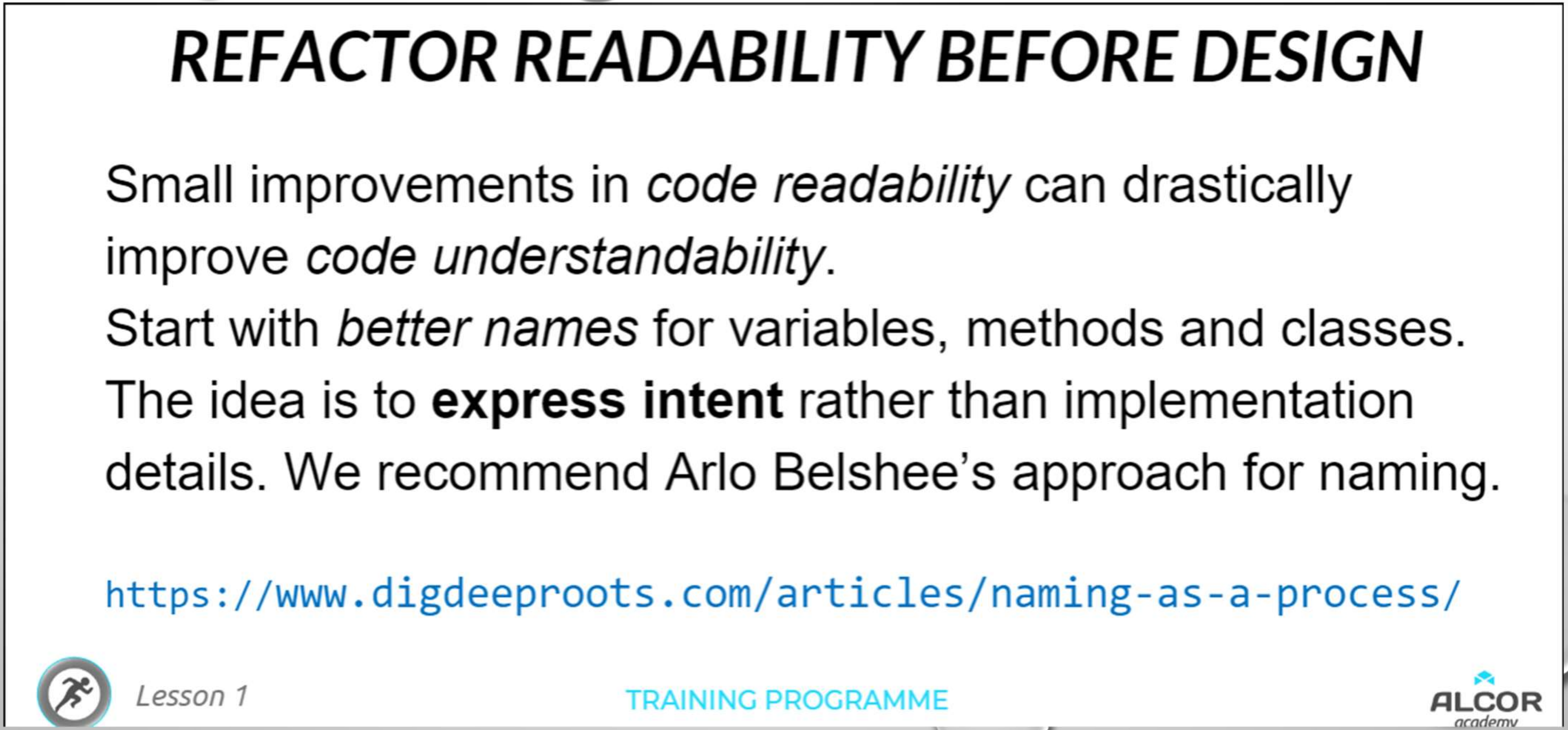
IL MONDO ERA COSÌ RECENTE, CHE MOLTE COSE ERANO PRIVE DI **NOME,**
E PER CITARLE BISOGNAVA INDICARLE COL DITO
(GABRIEL GARCÌA MÀRQUEZ)

NAMING AS A PROCESS

Arlo Belshee's approach for naming

Massimo Caccia

NOMINARE MALE LE COSE, È PARTECIPARE ALL'INFELICITÀ DEL MONDO
(ALBERT CAMUS)



REFACTOR READABILITY BEFORE DESIGN

Small improvements in *code readability* can drastically improve *code understandability*.
Start with *better names* for variables, methods and classes.
The idea is to **express intent** rather than implementation details. We recommend Arlo Belshee's approach for naming.

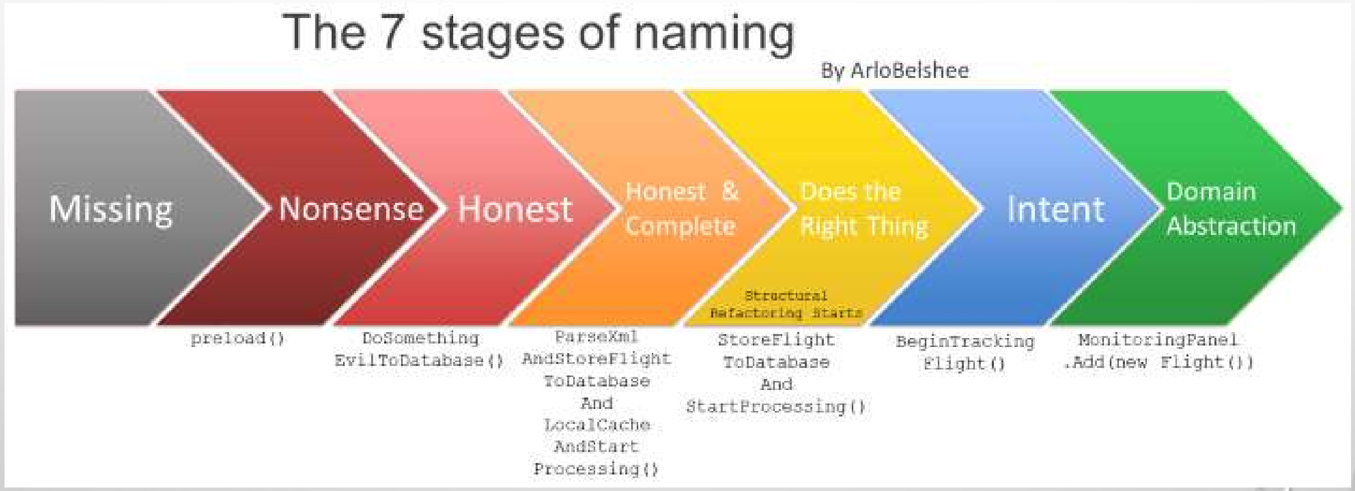
https://www.digdeeproots.com/articles/naming-as-a-process/

Lesson 1 TRAINING PROGRAMME ALCOR academy

GIOVANOTTO, I **NOMI** SONO COSE POTENTI
NON SI PUÒ USARLI IN GIRO SENZA NESSUN MOTIVO
(RICK RIORDAN)

NAMING IS A CONTINUOUS PROCESS

Often we try to find the perfect name on the first attempt, but actually naming is a process and Arlo Belshee's 7 stages of naming outline the concept of naming being a continuous process

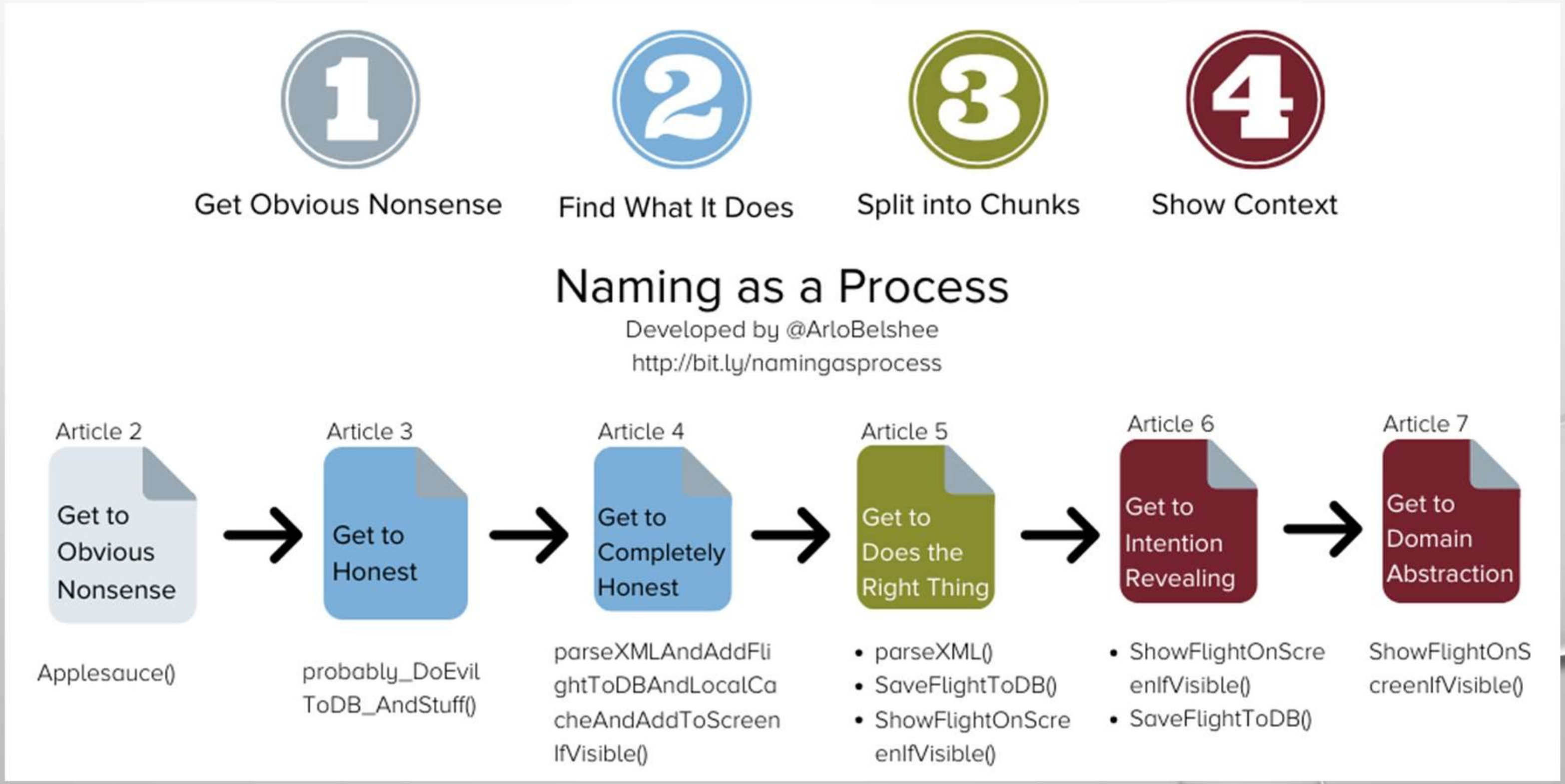


The 7 stages of naming
By ArloBelshee
Missing
Nonsense
Honest
Honest & Complete
Does the Right Thing
Intent
Domain Abstraction
Structural Refactoring Starts
preload()
DoSomething EvilToDatabase()
ParseXml AndStoreFlight ToDatabase And LocalCache AndStart Processing()
StoreFlight ToDatabase And StartProcessing()
BeginTracking Flight()
MonitoringPanel .Add(new Flight())

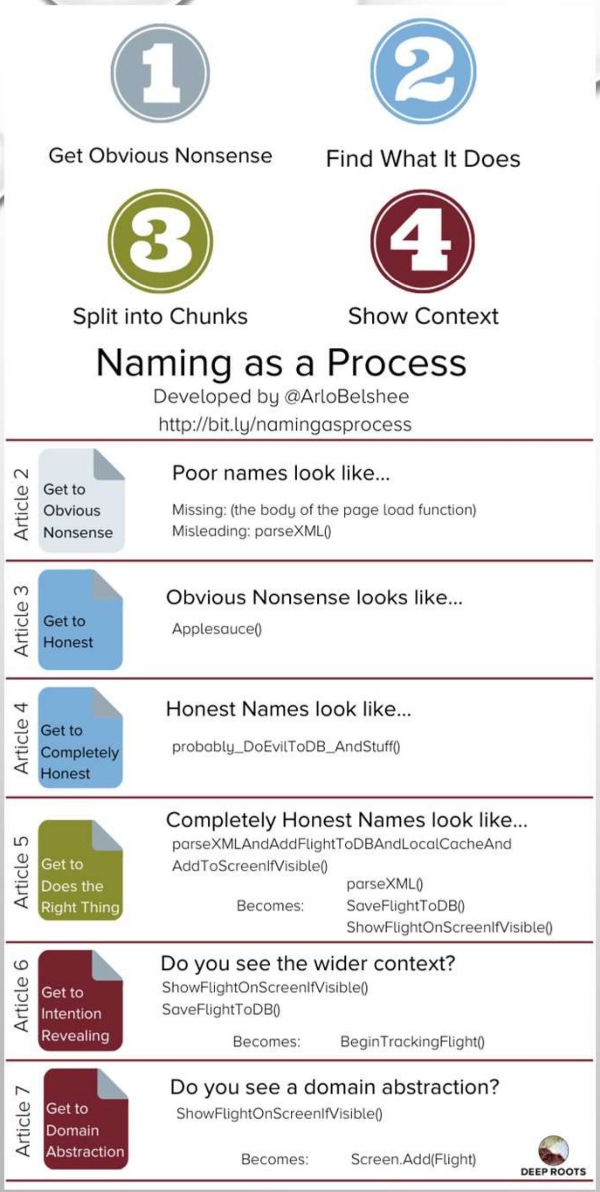
An important part of the development process is to ask **how the name can change, rather than why it shouldn't change.** Naming and refactoring is a continuous path, and this idea of the 7 stages highlights the importance of taking every opportunity you can to improve it. This in turn feeds into the idea of **slow change,** which is gradual change that may not be immediately obvious but overtime causes enormous change.

DIAMO UN **NOME** E DOPO NON SAPPIAMO
CHE **NOME** DARE AL **NOME**
(ANTONIO PORCHIA)

"Naming as a Process" highlights a way to extract domain knowledge out of the code and in to human readable form through the names of variables and methods.


1 Get Obvious Nonsense
2 Find What It Does
3 Split into Chunks
4 Show Context
Naming as a Process
Developed by @ArloBelshee
http://bit.ly/namingasprocess
Article 2 Get to Obvious Nonsense
Applesauce()
Article 3 Get to Honest
probably_DoEvilToDB_AndStuff()
Article 4 Get to Completely Honest
parseXMLAndAddFlightToDBAndLocalCacheAndAddToScreenIfVisible()
Article 5 Get to Does the Right Thing
• parseXML()
• SaveFlightToDB()
• ShowFlightOnScreenIfVisible()
Article 6 Get to Intention Revealing
• ShowFlightOnScreenIfVisible()
• SaveFlightToDB()
Article 7 Get to Domain Abstraction
ShowFlightOnScreenIfVisible()

DIAMO UN **NOME** E DOPO NON SAPPIAMO CHE **NOME** DARE AL **NOME** (ANTONIO PORCHIA)


1 Get Obvious Nonsense

2 Find What It Does

3 Split into Chunks

4 Show Context

Naming as a Process

Developed by @ArloBelshee

http://bit.ly/namingasprocess

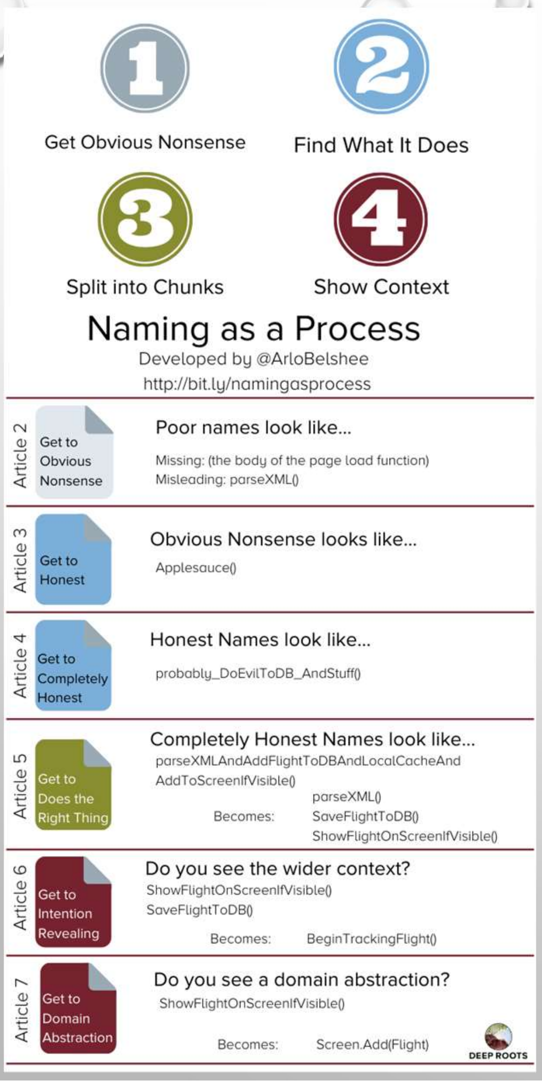
Article 2	Get to Obvious Nonsense	Poor names look like... Missing: (the body of the page load function) Misleading: parseXML()
Article 3	Get to Honest	Obvious Nonsense looks like... Applesauce()
Article 4	Get to Completely Honest	Honest Names look like... probably_DoEvilToDB_AndStuff()
Article 5	Get to Does the Right Thing	Completely Honest Names look like... parseXMLAndAddFlightToDBAndLocalCacheAndAddToScreenIfVisible() Becomes: parseXML() SaveFlightToDB() ShowFlightOnScreenIfVisible()
Article 6	Get to Intention Revealing	Do you see the wider context? ShowFlightOnScreenIfVisible() SaveFlightToDB() Becomes: BeginTrackingFlight()
Article 7	Get to Domain Abstraction	Do you see a domain abstraction? ShowFlightOnScreenIfVisible() Becomes: Screen.Add(Flight)

DEEP ROOTS

There is a suggested process for naming extracted chunks of code that aims to highlight any latent domain knowledge in a code block. The suggested steps for naming an extracted block of code or improve names are the following:

- Missing names to Nonsense names
- Nonsense names to Honest names
- Honest names to Honest and Complete names
- Honest and Complete names to names that Do the Right Thing
- Names that Do the Right Thing to names that show Intent
- Names that show Intent to names that form a Domain Abstraction

DIAMO UN **NOME** E DOPO NON SAPPIAMO CHE **NOME** DARE AL **NOME** (ANTONIO PORCHIA)


1 Get Obvious Nonsense
2 Find What It Does
3 Split into Chunks
4 Show Context

Naming as a Process
Developed by @ArloBelshee
http://bit.ly/namingasprocess

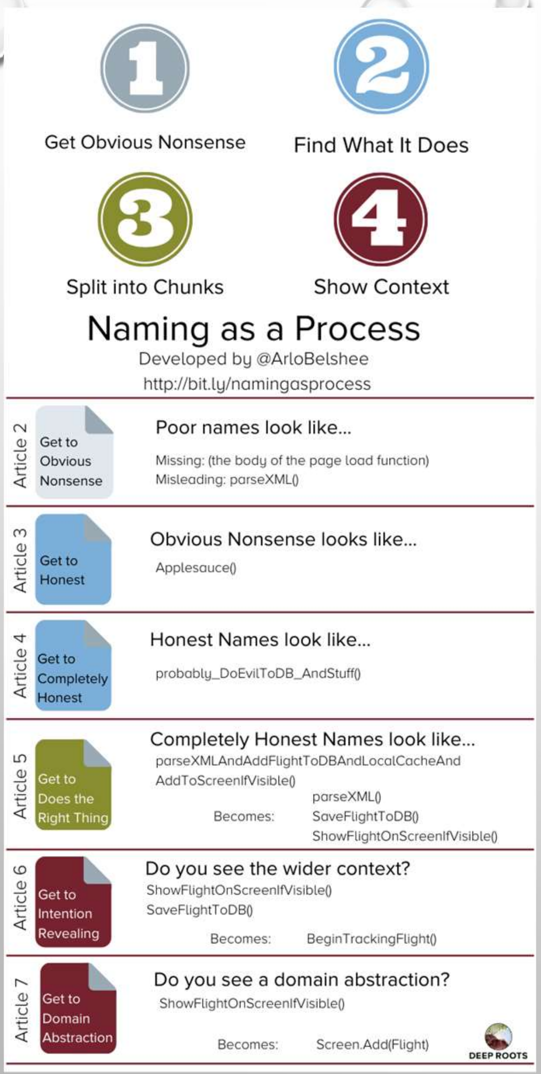
Article 2	Get to Obvious Nonsense	Poor names look like... Missing: (the body of the page load function) Misleading: parseXML()
Article 3	Get to Honest	Obvious Nonsense looks like... Applesauce()
Article 4	Get to Completely Honest	Honest Names look like... probably_DoEvilToDB_AndStuff()
Article 5	Get to Does the Right Thing	Completely Honest Names look like... parseXMLAndAddFlightToDBAndLocalCacheAndAddToScreenIfVisible() Becomes: parseXML() SaveFlightToDB() ShowFlightOnScreenIfVisible()
Article 6	Get to Intention Revealing	Do you see the wider context? ShowFlightOnScreenIfVisible() SaveFlightToDB() Becomes: BeginTrackingFlight()
Article 7	Get to Domain Abstraction	Do you see a domain abstraction? ShowFlightOnScreenIfVisible() Becomes: Screen.Add(Flight)

DEEP ROOTS

Nothing - You begin with nothing. There are no useful method names in the code… just a series of calls to other methods from constructed objects, conditionals, and a bunch of poorly named variables… At this point the code is in its least readable state, but you do have a hunch that certain blocks could be grouped together in some way and described more semantically.

Nonsense - You follow your hunch and extract a code block without knowing much about the ultimate function of the block in the program… At this point it doesn't make sense to waste time trying to think of a specific or relevant name, since the name is likely to change as you continue to develop understanding. You name the newly extracted method applesauce and move on.

DIAMO UN **NOME** E DOPO NON SAPPIAMO CHE **NOME** DARE AL **NOME** (ANTONIO PORCHIA)



1 Get Obvious Nonsense

2 Find What It Does

3 Split into Chunks

4 Show Context

Naming as a Process

Developed by @ArloBelshee

http://bit.ly/namingasprocess

Article 2 — Get to Obvious Nonsense — Poor names look like...
Missing: (the body of the page load function)
Misleading: parseXML()

Article 3 — Get to Honest — Obvious Nonsense looks like...
Applesauce()

Article 4 — Get to Completely Honest — Honest Names look like...
probably_DoEvilToDB_AndStuff()

Article 5 — Get to Does the Right Thing — Completely Honest Names look like...
parseXMLAndAddFlightToDBAndLocalCacheAndAddToScreenIfVisible()
Becomes: parseXML() SaveFlightToDB() ShowFlightOnScreenIfVisible()

Article 6 — Get to Intention Revealing — Do you see the wider context?
ShowFlightOnScreenIfVisible()
SaveFlightToDB()
Becomes: BeginTrackingFlight()

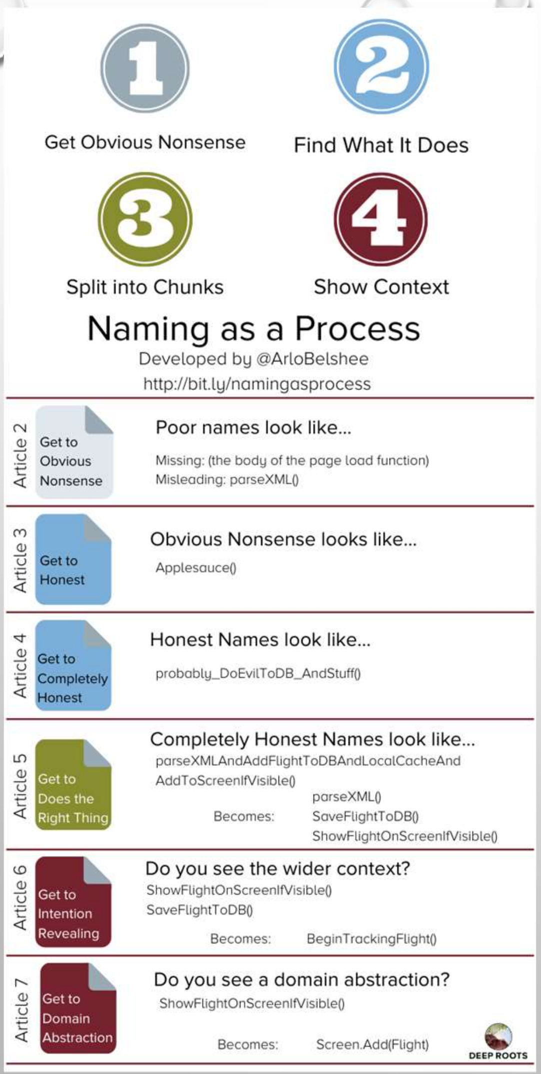
Article 7 — Get to Domain Abstraction — Do you see a domain abstraction?
ShowFlightOnScreenIfVisible()
Becomes: Screen.Add(Flight)

DEEP ROOTS

Honest - You begin to take a closer look at applesauce and realize that it is doing quite a few things: It seems to be operating on some object that contains user preferences, and makes a call to a payments service with user information. You update the method name to `GetUserPreferencesAndSendToPaymentService` since that is about as honest as you can be when figuring out what the heck is going on.

Honest and Complete - Upon further investigation, you realize the code is not just operating on the UserPreferences object, but also making a call to a payments service by constructing an intermediary Payments object, and it is writing to a database that contains invoices. You rename the function again to: `GetUserPreferencesAndConstructPaymentsObjectAndSendToPaymentServiceAndWriteToInvoiceDB` since this is the complete description of functionality in this code block.

DIAMO UN **NOME** E DOPO NON SAPPIAMO
CHE **NOME** DARE AL **NOME**
(ANTONIO PORCHIA)



1 Get Obvious Nonsense
2 Find What It Does
3 Split into Chunks
4 Show Context

Naming as a Process
Developed by @ArloBelshee
http://bit.ly/namingasprocess

Article 2	Get to Obvious Nonsense	Poor names look like... Missing: (the body of the page load function) Misleading: parseXML()
Article 3	Get to Honest	Obvious Nonsense looks like... Applesauce()
Article 4	Get to Completely Honest	Honest Names look like... probably_DoEvilToDB_AndStuff()
Article 5	Get to Does the Right Thing	Completely Honest Names look like... parseXMLAndAddFlightToDBAndLocalCacheAndAddToScreenIfVisible() Becomes: parseXML() SaveFlightToDB() ShowFlightOnScreenIfVisible()
Article 6	Get to Intention Revealing	Do you see the wider context? ShowFlightOnScreenIfVisible() SaveFlightToDB() Becomes: BeginTrackingFlight()
Article 7	Get to Domain Abstraction	Do you see a domain abstraction? ShowFlightOnScreenIfVisible() Becomes: Screen.Add(Flight)

DEEP ROOTS

Domain Knowledge - Whew that is one long method name! At this point, the complete description of the name is indicative that the code block is doing too much. A rule of thumb: methods should have a single responsibility, and the use of the word “And” in a complete method description is a red flag that the method should be broken up. A suggestion at this point is to break up the larger method along the “And” boundaries in its name, which will delegate one unique concern to each method. These method descriptions get you closer to latent domain knowledge stored within the original code: in this case, the sequence of steps to construct and write payments to the invoice DB!

BISOGNA SEMPRE CHIAMARE LE COSE CON IL LORO **NOME**. LA PAURA DEL **NOME** NON FA CHE AUMENTARE LA PAURA DELLA COSA STESSA (DAL FILM HARRY POTTER)

Conclusion

At this point, you haven't written any new code… You've just taken spaghetti code that previously existed, and have refactored it in a way that makes it more readable and makes the underlying domain knowledge more apparent to the reader.

Not only did it help you get a better understanding of a particular code block, but it will also prevent the next poor soul who needs to make a change in this block from going through the same process of understanding that you just went through.

You've effectively reduced the amount of time to comprehend the code for the next developer who passes through, which is unarguably valuable.

I GIORNI VENGONO DISTINTI FRA LORO,
MA LA NOTTE HA UN UNICO **NOME**.
(ELIAS CANETTI)

ANY QUESTION ?

Links:
https://www.digdeeproots.com/articles/on/naming-process/
https://www.digdeeproots.com/articles/naming-as-a-process/
https://www.digdeeproots.com/articles/get-to-obvious-nonsense/
https://www.digdeeproots.com/articles/get-to-honest/
https://www.digdeeproots.com/articles/get-to-completely-honest/
https://www.digdeeproots.com/articles/get-to-does-the-right-thing/
https://www.digdeeproots.com/articles/get-to-intent-revealing/
https://www.digdeeproots.com/articles/get-to-domain-abstraction/

https://bambielli.com/posts/2017-02-26-read-by-refactoring-pt-2/

https://softwareengineering.stackexchange.com/questions/404195/should-i-use-and-in-a-function-name

DARE UN **NOME** ALLE COSE È LA GRANDE E SERIA CONSOLAZIONE CONCESSA AGLI UMANI
(ELIAS CANETTI)

THANKS FOR YOUR ATTENTION !

Massimo Caccia
Software Development

Ente Ospedaliero Cantonale
Area ICT
Viale Stefano Franscini 4
CH-6500 Bellinzona

+41 (0)91 811 13 74
massimo.caccia@eoc.ch
www.eoc.ch