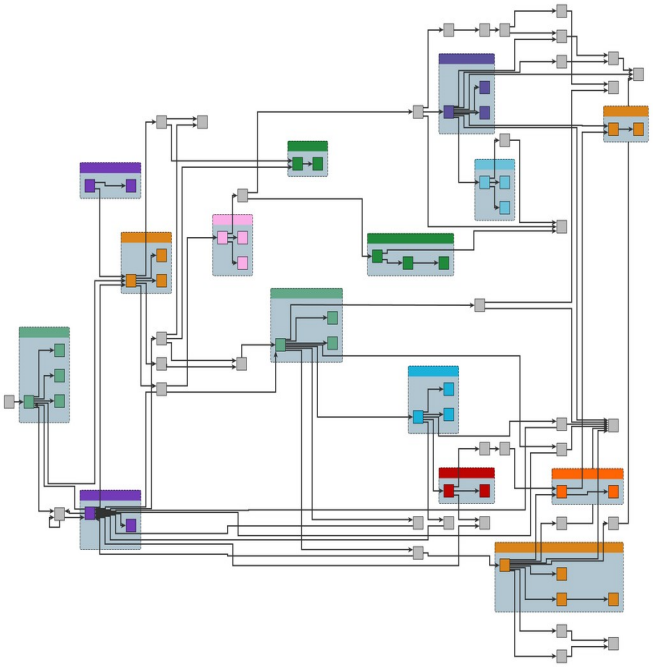
Eigenskills

Productive Programming

Productive Programming Course

Get the code to work *for you,* not against you.

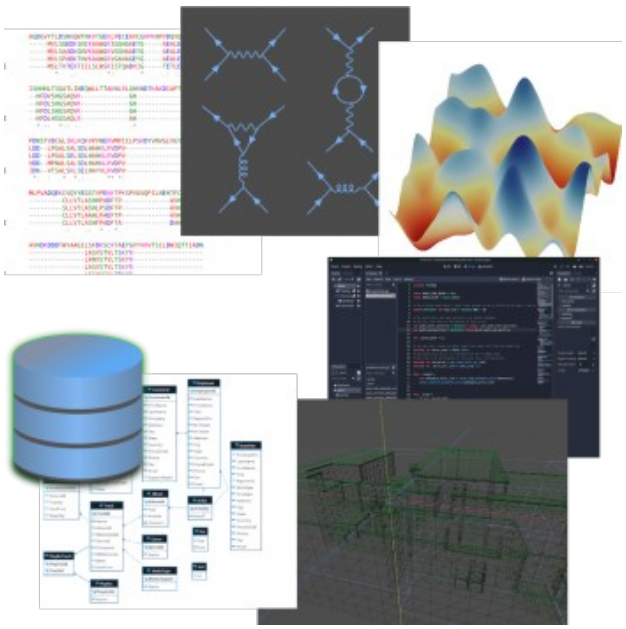
The software industry has decades of good practices and lessons learned when it comes to developing reliable and flexible programs.



But you don’t need a software engineering degree to pick up the core principles!
In fact, you don’t even need to be active in the ”traditional” software industry. The core principles are widely applicable and are a natural fit to domains ranging from scientific research to game development.

You also don’t need much experience with programming. If you’ve been coding for as little as 4 months, you can benefit from Productive Programming!

Picking up these practices will make things much easier for you as you grow as a programmer.



The Productive Programming course is designed to give you a shortcut to the ”wisdom” developers normally pick up gradually during their careers, often by diffusion from working in teams, by attending various workshops, or reading books. This latter material is often aimed at fellow specialists: tailored for its industry, dense, technical, and full of jargon. Productive Programming adapts the concepts and gives you a flying start to improving your work and learning more.

And this kind of knowledge is a great way to bridge career paths and find new opportunities. In fact, it can give you an advantage over the typical job candidate.

Think about it: Everybody’s collecting certifications. How do you stand out?
The *way* you work and approach problem solving is a good way.

When will you use this?

Productive Programming teaches a generally applicable approach to designing software, but some concrete cases where it provides significant value:

You frequently need to change the direction of your project.

This is actually much more common than you'd think, in e.g. science, startups and small business, game development.

There is no time for cleanup

Let's face it. We've all thought "*I'll polish this at the end*".

This basically never happens. You're 1 week away from submitting the paper – there is no time to clean up. And 2 weeks from now you'll be working against the next deadline. But by following good, simple practices along the way, you'll have your code in a usable, shareable state.

You absolutely must avoid mistakes.

Not only should scientific software be error-free, but for critical systems in medicine, aviation, and infrastructure, bugs can be life-threatening. It's not just that the programmers in question are very careful, they follow good patterns and sound methodologies.

Your users expect the software to work reliably.

This should hopefully not be too exotic :)

Productive Skills

You will learn patterns for:

- structuring your code
- proactively minimizing error
- delivering working software
- ways of working

These patterns apply to all levels and are relevant for projects of any size. Even those "throw-away" prototypes. More often than not, the "final product" is the prototype with more duct tape and paint. So why not build something good from the start?

The Productive Programming principles can be applied incrementally, without much overhead, but the savings down the line are well worth it!

Also, by working smart, you can avoid the stress and grind of those long work hours of wrestling your programs.

Classes

The emphasis is on practical use, so you will have plenty of exercise opportunities, both in pairs and individually. And you just need a browser! An online platform will be provided.

The groups are kept small, to give you more individual feedback. You can also bring your own code! Are you struggling with how to structure a certain script? Do you feel like you're constantly hunting bugs in it? Try out the Productive Programming patterns! Your code will only be shared with the instructor so you can get guidance.

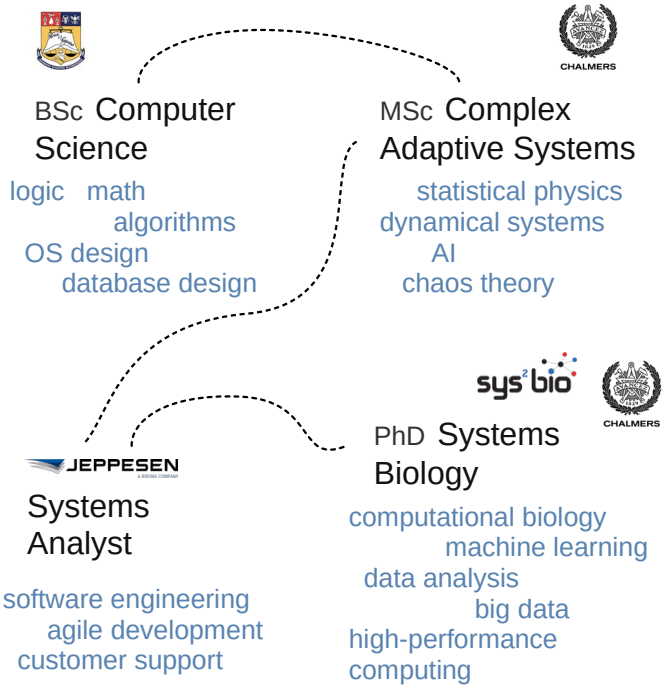
The philosophy of the course is to address individual needs and experience. Exercise material and examples will be supplemented accordingly. The instructor is familiar with many languages, so don't feel intimidated if you don't know Python. Examples are kept conceptual and use simple language. Get in touch if you're unsure or wish to find out more!

The Instructor

Filip has abundant experience with both scientific and professional software development. He has a PhD in systems biology, a Master's in complex adaptive systems, and a Bachelor's in computer science. As a professional software developer, he has worked for Jeppesen Systems on flight scheduling software.

His programming experience spans over 25 years, working with systems and projects of varying sizes and lifespans. He has also played a range of roles, from high level design, modeling, analysis, to project coordination and customer support, to building software pipelines from scratch.

Areas he has worked in include computational biology, bioinformatics, machine learning, big data processing, designing algorithms, data structures and software architectures, knowledge engineering and artificial intelligence, constraint programing, dynamical and chaotic systems simulation and analysis, specification systems, model checking and simulation, optimization, and network theory.



His teaching experience covers personal mentoring, undergraduate level courses, as well as workplace onboarding and training. He is a strong proponent of agile methodologies for both industry and science, and has a pragmatic and quality-oriented philosophy to software craftsmanship.

October 31, 2022

Links

Course page:	www.eigenskills.se/courses/prod-prog
Course schedule:	www.eigenskills.se/schedule
Contact:	www.eigenskills.se/contact

© Filip Buric, 2022. All rights reserved.
All logos and trademarks are the property of their owners.