A GUIDE TO FIND THE RIGHT BUSINESS MODEL FOR MATERIALS MODELLING SOFTWARE

EMMC White Paper

Gerhard Goldbeck, Alexandra Simperler;Goldbeck Consulting Ltd;
Natalia Konchakova, Daniel Höche;Helmholtz-Zentrum Geesthacht

August 2019

TABLE OF CONTENT

1. Overview

The objective of this White Paper is to help materials modelling software owners in exploring options to add to the existing universe of software, in particular to distribute their software more widely, to support the chemicals, materials and manufacturing industry with more tools, and potentially become entrepreneurs. It is based on evidence (Chapter 2) gathered from a wide range of stakeholders, in particular different types of materials modelling software owners (SWOs) and software users (SWUs).

Foremost, the SWO has to consider if their software is ready for the market and Chapter 3 poses some questions to aid in their decision.

Chapter 4 explores the reasoning why their software should enter the market. CATWOE (Customers, Actors, Transformation, Worldview, Owner and Environmental Constraints), a tool from the field of Soft Systems Methodology, is harnessed to enable the SWO to analyse if their software can find customers and actors, i.e. people using the software, and lead to some transformation in their professional life. SWOs are also encouraged to reflect on the worldview of their stake holders, and to revisit their ownership to ensure they are entitled to enter the market without barriers. It is also pertinent to think about environmental constraints, which are posed by forces the SWO cannot influence.

Chapter 5 is focussed on the existing Materials Modelling Software market. The ways how to enter a market depend on whether the market and products exist or are brand new or a combination thereof. A SCAMPER (Substitute, Combine, Adapt/Adjust, Modify, Put to alternative use, Eliminate and Reverse/Rearrange) analysis can aid in finding ways how to do this. The materials modelling software is “tested” by analysing if it could substitute an existing product, combine with an existing product, adapt/adjust to a problem, be modified, put to alternative use, or requires some elimination of code or rearrangement thereof.

Chapter 6 discusses ownership and licensing. The SWO has to either own the software or get permission by (co-)owners to move it towards a business. The importance of licensing is emphasised, as it needs to be clear what an SWU can/cannot do with the software.

Chapter 7 is dedicated to open source software. For SWOs interested in that route, it is important to form a community, as they require voluntary input for code development, documentation, maintenance and support. Suggested ways of generating revenue around open source software include research grants, collaborations in general and the offering of services. The latter can be providing training or Software as a Service (SaaS).

Chapter 8 elaborates on software commercialisation. Some SWOs may want to keep their academic careers and partner with a commercial software vendor or become a commercial software vendor. A commercial venture can be run as non-profit and for-profit organisation, and both will be discussed.

2. Evidence basis

2.1 Business Models for Materials Modelling Software Workshop

A workshop focussed on six software owners sharing their experience was held from 20th-21st of May, 2019 in Cambridge. The SWOs came from the UK, Germany and Austria and the participants from Belgium, Germany, Greece, Spain, and UK.

The business models discussed were

- Academic and Industrial Partnerships by Prof Mike Payne, CASTEP and University of Cambridge
- Forming a synergy between academia and industry by Dr Judith Rommel, Independent Expert
- Open Source Software by Dr Iain Bethune, STFC
- Non-profit organisation by Dr Georg Schmitz, Access e. V.
- From Academia to Business by Dr Amit Bhave, CMCL Innovations
- Academia and Business "One Model suits them all?", by Prof Thomas Schrefl, Danube University Krems and SuessCo

The speakers demonstrated different business models for materials modelling software and discussed with experts the relevant questions on market understanding, code functionality and new features development, training provision, software documentation requirements, specification and standardisation of software as well as licensing schemes and ownership.

2.2 EMMC International Workshop 2019

The EMMC International Workshop 2019 was held in Vienna from 25th to 27th of February 2019 and was attended by more than 170 European and international experts to discuss and contribute in setting a common direction among stakeholders in all areas of/related to Materials Modelling. Three sessions were dedicated to SWOs and their business approaches. The sessions covered progress in the industrial deployment of materials modelling software drawing on ample expertise of software owners, the issue of business models for materials modelling software and the complicated topic of the synergy and conflict between open source and commercial software for industrial deployment of materials modelling software.

2.3 Whitepaper: Business Models and Sustainability for Materials Modelling Software

This EMMC White Paper (Goldbeck & Simperler, 2019) is based on input received in a workshop with SWOs and surveys of SWOs and SWUs carried out online by the EMMC during 2018.

Successful software especially for materials modelling has an expected lifetime of many decades. This long-term nature requires a sound legal and business foundation: the ownership of software must be clearly established and the license models need to be carefully thought through to ensure a sustainable development and maintenance of the software and impactful exploitation by both academic and industrial

end-users. Different business models carefully need to be considered when developing a strategy for long-term sustainability of software and sustainment of the operation.

As background, the White Paper provides an overview of materials modelling software market, considering different segmentations and briefly discusses market dynamics.

Furthermore, the fundamental aspects of software sustainability and sustainment are described. Business models are discussed, with a detailed analysis of different revenue models. Emerging business models including digital marketplaces are considered.

The status of materials modelling software is presented with respect to different sustainment attributes (Users and Communities, Product Management, Software Development and Maintenance, Revenue Generation). Finally, the thoughts and recommendations shared by Software Owners during the feedback gathering are summarised.

3. Is your Software ready to enter the Materials Modelling Market?

According to common understanding and agreement within the community, software is ready to enter a market when the following questions have been answered satisfactorily:

- The ownership (i.e. copyright) of the code base is known and documented.
- All owners agree on making the code available to others.
- All SWOs have agreed on a licensing model.
- There is a clear responsibility for the release and for the maintenance (e.g. bug fixing) of the software.
- There are systems and procedures in place to ensure the software is as bug free as possible, tested on all platforms it should run on and its installation works faultlessly.
- The software should be robust and reliable.
- The latest hardware developments are taken into account and the software can be deployed on High Performance Computing platforms, as is widely expected in materials modelling.
- The software produces the expected results within statistical error margins (verification).
- The software should be validated for important cases and the validation should be well documented including the expected accuracy, so that users can reproduce results.
- The software should address important and pervasive materials research and application problems, most importantly including those that are of interest to industrial research. Software developers should be aware that industry is mainly interested in applications and production, whereas their original development of materials modelling software was rooted rather in theoretical foundations and fundamental research. To ensure that the software product is useful for industrial end users, the software provider should focus on the ability to model industrial materials and manufacturing challenges.
- The software should be easy to use and fit into existing workflows (e.g. industry workflows and important research workflows) and should come with a manual and documentation. It should be interoperable with relevant environments and platforms and seek to adhere to existing and emerging interoperability standards. For visualisation and pre-/ post-processing, either the required tools or interfaces to other tools that can do the job need to be provided.

Some helpful technical guide is available here: https://software.ac.uk/resources/guides/ready-release

4. Why should your Software be out in the open?

You may have developed your materials modelling software because there was nothing available to aid with your research. The supply and demand may have been limited to yourself and your research group. Your next task will be to find out, if there are other people who have a demand for your software.

In this case, several techniques can aid you. One is the **CAT**WOE tool (Smyth & Checkland, 1976). CATWOE is a Soft Systems Methodology (SSM) (Bergvall-Kareborn, Mirijamdotter, & Basden, 2004), and is used in information system development. **CAT**WOE is an acronym comprised of **C**ustomers, **A**ctors, **T**ransformation, **W**orldview, **O**wner, and **E**nvironmental Constraints, see Figure 1.

Figure 1. The six different perspectives of a CATWOE analysis: Customers, Actors, Transformation, Worldview, Owner, and Environmental Constraints.

4.1 Customers

These are the people who purchase or acquire your software. They can be an organisation (macro-level) or the actual user (micro-level) of your software. You may sell at different levels – to a person, a department, or on a corporate level; hence you are well advised to analyse carefully who your customers really are. If your customer is a person, try to gather success stories that will aid to sell to the next level within the same organisation or to the same level within another organisation. If your customer is indeed an organisation, you should analyse how they can benefit from your software. Also, your customers can be stakeholders that are not involved in the actual purchase but are affected by your software. For example, the IT personnel is

affected, because they will have to install your software. The line manager of your buyer might be affected, because they agreed to the purchase and are accountable for it.

4.2 Actor

Actors are the persons who will perform the simulations (here referred to as 'transformation') and who will work with your software. What actors do you require? Can they easily use your software? Do they require training? Do they need to be expert users? Who, do you imagine, will be able to use your software?

4.3 Transformation

Your software will take an input and transform it into an output. You have to ask yourself, how simple this transformation is. Are many steps required and can actors easily perform this transformation by themselves? Finally, you have to ask yourself, if actors are using your software, how will it benefit the customers? The actor may be pleased if your software produces e.g. band gap energies, but what would this mean to the whole organisation? Moreover, the preparation of new features / functionalities to achieve better and novel transformations will require time and money. Bear in mind, if actors cannot realise this by themselves or with support, you have to advise your customer what sort of training or consulting is involved.

4.4 Worldview

This allows you to reflect on the bigger picture your customers find themselves in. Organisations have to follow regulations, they compete with other organisations, etc. – could your software also benefit here? You should gain some knowledge of the domain including the industry around it. For example, if you offer software that could calculate materials properties of semiconductors, you should know what problems the industry is facing so you can offer a relevant solution. You can gain this understanding by attending fairs where members of this industry are likely to be present, and carefully listen. Always start with listening to the problems and hold back on offering (perhaps wrong) solutions. This is how you can change your worldview.

4.5 Owner

The owner of the software is maybe you, but it is important to ensure proper copyright ownership or rights to use for all parts of the software. You can learn more about this in Chapter 6, dedicated to ownership. It is important that a customer clearly knows who owns the software, as they want to license it from you and be legally entitled to use it. They will also want to know who is liable for your software to function correctly. Or they simply may want to come back to you and ask you to change the transformation or to add a new one, i.e. add new functionalities.

4.6 Environmental Constraints

There may be limitations to deploy your software, which are out of your control and are not part of your software. These could be export controls. There could be financial constraints if the Customer has no budget to purchase or licence your software. Organisations may have to go through 3rd party funding processes

which require paperwork and funding may only be awarded at particular times. Companies also may have to work to strict guidelines so your software has to undergo a period of validation. In this case you want to assure that there are synthetic data you can use to aid with a validation and thus, you can accelerate the process. Another important constraint is industrial confidentiality that can impede you to use real data to demonstrate your software's capability. Prepare synthetic data for demonstrations of the maturity of your software as well as for training materials.

4.7 Conclusions, Remarks

CATWOE is not a guarantee to success but can help you to see what your software business aims to achieve from six different perspectives. You may say your software should be out in the open because you developed already a notable userbase and people keep asking for more features. The latter is probably why you are looking into business models to finance their development. It is still worth reflecting with the CATWOE tool as you can uncover six different perspectives which may provide valuable insight for developing your business model, your go to market strategy or grant application. Public funding for pure software development is hard to come by, but if after careful stakeholder analysis you could identify R&D roadmaps, key industry sectors, worldviews, … the "transformation" you afterwards can offer via your software could be an innovative way to impact.

5. Get to know the Materials Modelling software market

The good news is that materials modelling and directly linked process modelling already plays an important role in a large variety of industries comprising automotive, aerospace, chemicals, consumer packaged goods (home and personal care etc), energy, metals and alloys, and electronics/semiconductors, specialty materials and pharmaceuticals. A number of key aspects are covered in the Goldbeck Consulting report on the scientific software industry (Goldbeck G. , 2017), which looks directly into materials modelling software markets. The relevant established market areas that are covered by analysts tend to be Cheminformatics (e.g. https://www.grandviewresearch.com/press-release/global-chemoinformatics-market), which is however mainly focussed on drug discovery but includes players also active in software for discrete (electronic/atomistic/mesoscopic) materials modelling, and Computer-Aided Engineering (CAE), with some examples of reports here:

- https://www.transparencymarketresearch.com/computer-aided-engineering-market.html
- https://www.businesswire.com/news/home/20171020005370/en/Global-CAE-Market-2017-2021-Growth-Application-Specific-CAE
- https://www.zionmarketresearch.com/news/computer-aided-engineering-market

The Continuum Materials Modelling Market is part of the CAE market and estimated in the region of at least €1bn. However, it is difficult to estimate the share of the market which is actually due to the use of continuum modelling methods (such as FEM or CFD related) for materials modelling rather than engineering modelling purposes. Broadly speaking we draw the distinction that materials modelling is involved in the design and detailed understanding of the material, whereas engineering modelling designs products 'with' the material, i.e. uses known materials properties for the actual materials description.

In addition to standard continuum mechanics and fluid dynamics methods there is a range of continuum models and related software which is about the detailed description and design of the material itself. These methods may provide a strong link into computer-aided engineering and hence are also likely to have a relatively high market potential.

Examples include Phase Field approaches, composite materials modelling based on either mean-field (homogenisation) or explicit microstructure mechanics modelling with FEM solvers, electronics and electromagnetism applications. Basically, these are methods that are most closely linked to supporting the later stages of the innovation process.

If your software serves the discrete modelling market you operate in a market size in the region of € 100m, i.e. an order of magnitude smaller than the continuum modelling market. There is evidence based on the number of users and publications (Goldbeck G. , 2012) that electronic models make up the largest part, followed by atomistic and then mesoscopic modelling. You have to learn who the main players are and see if there is a niche for you to fit in.

At the moment, software that can handle or contribute to solving multi-scale and multi-physics problems has high demand on the Materials Modelling Market. Additionally, development of unique algorithms and combination of physics-based modelling with the AI/ML data-driven approach would help you to find your niche in the modern materials modelling software market.

It is important at an early stage to check if the market has been already penetrated by other persons using your software. These could be academics/consultants using your software or even just results and provide these or services to your potential customers. This can happen, if the software ownership and/or licensing was not clear from the beginning. Chapter 6 offers some solutions on these topics. If your software is free and open source[1], third parties can legally use your software to offer services. In this case, you have to either find markets that have not been entered yet or develop new software functionalities and protect them with a suitable license model. But even if the market is penetrated, you just may have specific expertise in a niche area that has not been covered fully by another provider – evidenced by the fact that we find many relatively similar software codes offered by many SMEs.

Another important “market” consideration is the number of users in the field. Depending on your area it may be a relatively small number or a very large number. Examples are the field of (electro)magnetic devices [2] where it is quite possible to more or less know all users via conferences/meetings and close collaborations. In the field of electronic modelling using DFT codes the number of users has been estimated to be in the 10,000s, while in CFD it will be 100,000s. What could help here is an attempt to democratise your software, i.e. thinking about how your software can be made available to more people (Hanna & Weinhold, 2017). Typical of Materials Modelling Software is that it is never sold or offered without substantial support, training and services of some sort which can reach the advanced services of a training or consulting team. The latter could aid in penetrating the market as you can design your services to appeal to different user expert levels.

[1] Open source here is used in the sense of Free Open Source Software (FOSS), and refers to software that is licensed to be free to use, modify, and distribute.

[2] For example, http://www.suessco.com/simulations/ and http://www.qwed.eu/

Thus, if the market is saturated on the expert level, what could you do to reach the non-experts? Service provision can also help to retain your market, and a number of materials modelling software owners would never provide even trial licenses without training.

You could use the Ansoff Matrix (Ansoff, 1957), depicted in Figure 2, to check if your ideas are viable.

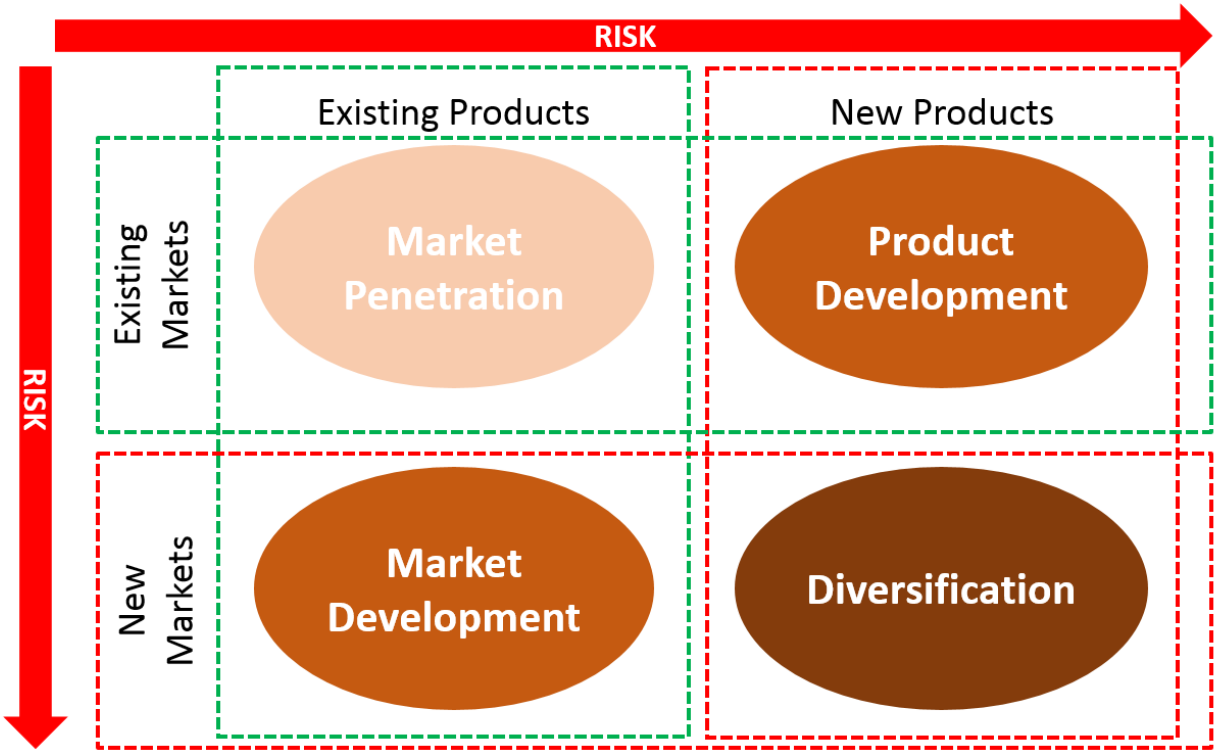

Figure 2. Ansoff Matrix

This matrix serves as a framework for decision making about strategies for bringing your software to the market. If the market and products exist, you need to find ways for your software to penetrate the market. You can do this with pricing or by offering better services (maybe already approved) around your software. If you think your software can do something novel, and you have detailed insights into customers' needs and how they change, you did product development and enter an existing market. You will need to convince the customers why they should change and adopt your software.

If your software is alike to existing products you could try to enter new markets, and do market development. Are there persons in the industry that do not use software, but yours has special features that can enable them? Do you have a novel offering say SaaS or user-friendly APPs? Diversification can be the way forward if both markets and products are new. But in this case, you take a high risk as you go into markets that are not known yet with software that was never tried.

Another technique that can make you think outside the box and create opportunities for your software could be SCAMPER (Eberle, 1972). SCAMPER is anacronym made out of **S**ubstitute, **C**ombine, **A**dapt/Adjust, **M**odify, **P**ut to alternative use, **E**liminate and **R**everse/Rearrange. The idea is you want to get away from usual solutions and generate new ideas. SCAMPER for SWOs can be:

- Substitute: Can my software be swapped with another existing one? Be careful here – from the customer's perspective the cost of switching to other software is high due to license fees, learning barriers, software infrastructure and integration. It will be up to you to make this substitution palatable.
- Combine: Can I combine my software with another one and create something new?
- Adapt/Adjust: Could your software save another purpose than you intended? What do you need to adapt/adjust? Could your software be adapted to serve non-expert users?
- Modify: What could you emphasise in your software to create more value? Could you modify an element within your software to create something new? Find unique things for your software, that competitors cannot do.
- Put to alternative use: Could another industry/domain profit from your software?
- Eliminate: Less is more – could you simplify/streamline your tool? If it would be less complicated, is it more attractive to users?
- Reverse/Rearrange: Cloud you offer the opposite/something completely different to what your software does? Could you rearrange the workflow you offer?

Finally, try to create opportunities by "meeting your market" at conferences, fairs, etc. and think where your users are most likely to be. If you stick to materials modelling conferences you tend to meet likeminded peers but you will not be confronted with your industry's problems. You can put a slide into your talks. You show some applications and formulate quite clearly, what your software can do in a language your future customers can clearly understand. It is important to prepare focused examples for the industrial audience of a conference and try to explain the economic benefits when using your software.

Chapter 7 will offer some more detailed insight into business models, but before thinking about these, you want to establish who really owns your software, which we will discuss below.

6. Software ownership

It is vital to establish before you write software to whom it belongs. If you work for yourself with your own equipment, the software you write belongs to you. If you work for a university or a private company usually, your employer has a shared or a full ownership to software you write, as the latter may be even part of your work duties

It is pertinent that you carefully read up on employment laws in the country you work in. Also, check in your personal contract what you can or cannot do. Contracts may refer to Terms of Employment of an organisation and even if they are not explicitly stated on your contract, you may have to follow them as well.

The business models you will chose at a later stage will depend on this ownership.

6.1 Clarify Ownership

If you work at a university or at a governmental supported research institute, find persons working in technology transfer departments who should be able to aid you with commercialisation attempts and give you a fair outline on what return you can expect. Do this early on in your venture so you can decide if the

return on investment is worth your effort. The main thing you want to clarify is in case you generate an income with your software if royalties or parts of the income have to be paid to your host.

You could opt now for open source; however, even non-profit software should have a clear owner. You still want to protect your intellectual property and have some quality control over your brainchild.

If you work with colleagues or students together it also important to clarify ownership. It is beneficial to work out a joint ownership with your colleagues so everyone gets the right credit for their work. If the work on your software is performed by students, you may need their agreement that they pass on their work to your ownership when they finish their PhD, for example. Student contracts tend to be different to regular employment contracts, so you may want to check with your university what rights students have when they write code as part of their MSc or PhD and if there are formalities to hand these rights over to you.

For example, a number of development teams formed foundations such that ownership rights are always transferred to the foundation. Examples are Quantum Espresso[3] and CP2K[4]. This can be very helpful if developers work for different institutions and want to declare common ownership.

If you work in industry, but think of becoming a materials modelling software entrepreneur, you may want to check your contract and see, what the right time is to embark on your new career. Some employers may add clauses to a contract that prevent you from starting your new business right away to avoid competition. Your employer also may have the right to prevent you from “moon lighting”, i.e. working on software in your spare time.

Employment laws can be very complex and are different for each country, so it could be a good idea to invest in some legal advice.

6.2 Licensing

Licensing is a way to protect your intellectual property (IP) and even if you are an advocate of free and open source software you should consider a licence. A software license governs how the software can be used or redistributed. The most common licence types are proprietary licenses, permissive and copyleft. All three license types let the copyright owner retain the copyright.

Open source software owners often go for copyleft licenses (such as GPL and LGPL) which allow users to perform, display, copy and modify the software. These licences do no permit sublicensing and distribution is only permitted under the same license, which can cause problems for commercial ventures.

If you want to permit sublicensing, you could choose permissive licenses (such as BSD, MIT, Apache) which offers the best guarantee of downstream up-take of your software if needed.

Proprietary licenses are the most protective and permit a user only the right to perform and display. You have to take care when you develop your software as libraries and code you implement must not be

[3] http://foundation.quantum-espresso.org/objectives

[4] https://www.cp2k.org/foundation

copyleft licenced. You would then have to adopt the same copyleft licence and are unable to commercialise your software as intended.

If you are overwhelmed by these licence choices, you can find some help here:

- https://ufal.github.io/public-license-selector/, provided by The Institute of Formal and Applied Linguistics at Charles University in Prague
- https://choosealicense.com/licenses/

You need to take care when you license your software to a person or company. You will be liable for certain aspects of your software. Thus, the "limitation of liability" will be the one of the most important clauses in your software license agreement. You will have to state a limited amount and types of damages your customer can recover from you. For example, the GPL License v3 comprises:

16. Limitation of Liability "IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES." (taken from[5])

Liability clauses are sensible for open source software owners and a **must** for commercial entrepreneurs.

7. Business Models for Open Source Software

In respective societies the phrase "Open Access - Open Science" is very popular and making software open source permits people worldwide to use software free of charge. The philosophy of Free Open Source Software[6] means that users have the freedom to use it (for any purpose), to modify and to redistribute it.

A large user base/community/market makes is much more likely for open source software to be successful, e.g. DFT (NWChem)[7] or CFD (OpenFoam)[8]. The mechanisms of making the open source software sustainable could be by contributions to the code from a larger community (though that also requires management), the recognition and hence funding that the developer(s) get for serving such a large community (i.e. having big impact) and the opportunities to monetise/valorise the IP via services and add-ons.

However, open source software also needs sustaining maintenance and new functionalities, which requires funding. There are also business opportunities associated with open source software, as the well know example of Linux-OS showed.

[5] https://www.gnu.org/licenses/gpl-3.0.html

[6] https://opensource.org/osd

[7] http://www.nwchem-sw.org/index.php/Main_Page

[8] https://openfoam.org/

7.1 Form a community

Open source software in the materials modelling world is typically brought to life by a single person or a small group of like-minded people who could use their research time, spare time, some funding, and a great amount of enthusiasm. It is pertinent that the software development does not end up in a GitHub repository which is littered with non-updated software or some incoherent developments, as people tend to work on parts of code that are most dear to them. If you are the developer, you have to take charge so this does not happen and aim to establish and maintain a community of developers.

For example, DAMASK — the Düsseldorf Advanced Material Simulation Kit (GPLv3 licence), offers guidelines for their contributors to make sure the development remains professional, see [9].

You should also generate platforms where people can meet virtually and discuss your code, share advice, pose questions, etc. For example, Mattermost[10], google groups[11], or similar forum types are very useful. You (and your developer colleagues) will have to be very active to get this started and populate such a forum with contents as well as moderate the forum. Users need to see content, and the forum should appear dynamic and active; otherwise they do not come back. Invite some expert users of your software to take part in answering questions, sharing tutorials, writing some documentation until you see other users come forward and take part. You should also offer a code of conduct, as you want every forum participant to feel welcome. It would also be beneficial to have a section where users could enter which functionality they would like to see in the future.

As mentioned above, to preserve and protect their software some open source developers have formed Foundations, such as afore mentioned Quantum Espresso and CP2K. You may want to get legal advice in how to set this up and also what it takes that your university passes on its IP (if they own it) of your software to this foundation.

7.2 Creating Revenue around Open Source Software

7.2.1 Research Grants

It is often possible to get research grants for developing new features but these grants are often relatively short. Thus, it is in your interest to keep an application pipeline going. Besides new features, you also want to improve your code and make it sustainable. It is still nearly impossible to find grants solely dedicated to improve software. The Software sustainability Institute made a start to compile some sources which you can find here: https://www.software.ac.uk/how-fund-research-software-development. The Partnership for Advanced Computing in Europe (PRACE[12]) may also have some useful grants.

[9] https://damask.mpie.de/Development/WebHome

[10] https://mattermost.com/

[11] https://groups.google.com

[12] http://www.prace-ri.eu/

One successful strategy is to apply for funding to add new features to your code that address new science and at the same time factor in some time to do code maintenance and improvements. The argument you offer is, that this "extra" creates a better, validated and robust tool, which will aid to find innovative materials or processes faster and thus, will have greater impact.

7.2.2 Collaborations

You could consider to collaborate with a commercial partner and have them pay for your time and effort to develop new functionality. The commercial partner will often ask you to have exclusive rights on the software for some time before you can make it available to everyone.

An interesting collaborative program in the UK is "Innovation Return on Research" (IROR[13]) The programme develops a suite of digital assets, which are industry relevant in the areas of Chemistry and Materials, Life Sciences, Engineering and Manufacturing and Enabling Technologies.

In general, there may be opportunities to collaborate with a national HPC centre or a governmental innovation centre, and offer your software as a digital asset to businesses. These businesses and you should find a common interest, validate and shape the capabilities of your software.

7.3 Offering Services

Offering services such as training or contract research/consulting or software as a service can create revenue for developing your software. There is an interest of companies to use cutting-edge science in their research, which opens a decent market. However, services require time and effort at your end. You also have to check, in case you are an employee, if your employer agrees with you doing this. Universities, for example, allow a certain amount of days you can do consultancy or similar. If you have decided to dedicate your career to make a living from your open source code, then offering services will be an important income stream.

7.3.1 Training

Training is a good way to grow your userbase but it needs materials and it takes time to develop them. It would be a good idea to codevelop these with your superusers and your community. Good training materials comprise a mix of presentations and tutorials. Some SWOs discovered YouTube as a medium to post training videos. These mostly are comprised of screen shots and audio. Also training materials should be licensed and we recommend the creative common licenses[14]. These licenses allow giving credit to the creator of training materials and you can decide what you permit, such as changes, mix and match, distribution, etc.

You can do online training and choose free platform and VoIP solutions to work cost free and thus, reach out to your community. You should either cater for different time zones or record the training to make sure you reach your audience worldwide.

[13] https://www.hartree.stfc.ac.uk/Pages/IROR.aspx

[14] https://creativecommons.org/licenses/

Some academic Open Source SWOs offer live training workshops and find a host institution that can offer facilities. Usually, the participants are charged a fee to cover the costs and are responsible for their travel and expenses. Your costs will be the training venue rent, your travel and expenses, and food and coffee breaks for the participants. It can be worthwhile to check if a hosting institution has budget to co-finance such events or you have a commercial sponsor. You also could offer training services to commercial users of your software.

Some SWOs offer between two to five days onsite training. Industry prefers two days as this is the optimal time an employee can stay away from their day-to-day duties. They also want you to focus on application of the software rather than the theory behind and they would expect you to train them on examples relevant to their work and business. Academia is often interested in five days events as they also expect intensive lectures and tutorial blocks as well as a free-styles modelling part on user focused examples.

7.3.2 Contract Research or consulting

You can offer to do research for a customer, i.e. doing the work with your software for them, or consult them on how to tackle a problem using your software. Usually you have the choice to do “official” or “private” consulting (if permitted). If you consult officially, you would work via your university consultancy service and they set up contracts, remuneration, etc. for you. If you decide to consult privately, your university does not cover any liabilities, will not permit you to use university affiliation, equipment, and staff. Thus, your customer must have the means to run your software on their premises.

As a commercial Open Source SWO you should draft a contract with your customers and contents may be:

- Remuneration, timeframes, trade secrets
- Statement of Work (SoW), project scope
- Appropriate indemnities
- Services and deliverables with a statement: IP in the deliverables and the services will transfer to the company immediately upon creation (or not)
- All materials created during the consultancy have to be handed over and destroyed at the consultant’s premises and from their devices (or not)

During each consultancy, you will gain know-how. Thus, you should state also:

- Know-how is excluded from the definition of IP, since you cannot “unlearn” it
- You will have the right to use your new know-how for the provision of similar services to any other clients

The legislation surrounding consulting and the actual wording may be different in your country, so do check carefully or employ some legal aid.

7.3.3 Software as a Service (SaaS)

We will define SaaS based on software that is owned, delivered and managed by you and is available remotely to customers/users. You offer access to the software and the hardware it is running on, i.e. you offer the infrastructure and support that this infrastructure is running faultlessly. This requires you to provide

hardware and the obvious thing is to offer access via the cloud. Some customers still have concerns about how secure the cloud is with respect to the data. Also, this requires investment from your end as you have to procure the actual hardware and make sure all is running 24/7.

7.3.4 Running a business based on Open Source Software

As we have seen above, there are a range of opportunities to create revenue from open source software, including

- Direct collaboration agreements with industry to develop new functionality in the code. Note that Open source licenses do not oblige the owner of new developments to disclose them. Hence, the right to disclose can be given to a commercial partner, typically retaining the code confidential for a period of time. Some companies also include a second tier of members of a consortium that pay to be next in getting access for a further limited period, after which the code enters the general open source library.
- Closed source, proprietary licensed 'add-ons'. Depending on whether a code is copy-right or copy-left, there may be limitations. A typical example that works in most cases is to develop materials relations (e.g. forcefield for atomistic modelling) with high industrial value (e.g. well validated for a class of materials important to a particular industry).
- Services including preferential bug-fixing and maintenance services as well as training, contract research and consulting.
- SaaS, which typically includes some easy to use web application.
- Graphical user interfaces with proprietary licences.

All of the above have been applied successfully in materials modelling, typically with a mixture of revenue streams, and often involving a mixture of individual customer relations and consortia.

8. Business Models for proprietary software

Your business model will depend on whether you wish to pursue an academic career or become a commercial owner and thrive in the business world. The following cases are most common:

8.1 Business Models for an Academic SWO

8.1.1 Academic career and partnering with a commercial software vendor

Before you embark on contacting potential software vendors you should get some advice on licensing e.g. by working with your University technology transfer office. You should always protect your software in some way and keep track on how widely it is used. Be aware that licensing negotiation can take time until you find the right agreement and change of wording may go back and forth between lawyers. Experienced academics state 1.5 years as a minimum before a licence agreement can be found.

You should in any case ensure that your software is robust and easy to use and developed to a high software engineering standard. The latter makes it more adaptable to industry, especially if your software should be part of a commercial software suite. During development you want to make sure you do not use GPL (or related licenses) code or libraries as this may turn your whole software into GPL and make it less suited for a commercial vendor to include in their suite.

Check out several companies and explore how their vision relates to yours. Find out if a product manager is there to oversee the materials modelling software as this is vital for a good overall product and thus, better sales.

Once you opt for a company the negotiations will start and you want to negotiate royalties. Ideally, negotiate a fixed upfront payment and in addition a royalty on sales ideally with a guaranteed minimum annual payment. This guarantees an income stream and you can reliably fund the development of your software. Generally, the better and more usable the software (marketable product) the higher the royalty you can negotiate is. Also, royalty rates vary depending on how much effort you commit in terms of maintenance and adding new functionalities. However, you cannot expect a 50% royalty as marketing and support cost money. More typically, royalty rates will be in the 10-20% range as over the lifetime of a commercial software, development is about 20%, while marketing, sales, support, and training require 80% of the resources. This is also a point where you want to involve experts in such negotiations, e.g. from your University Technology Transfer Office, Funding Agencies, Angel Investors, other Entrepreneurs, etc.

Having a good estimate of market size and confidence in the potential sales of your code by the vendor will be important as low sales and royalty payments may mean both sides loose interest quickly. For example, assuming licenses worth €100,000 are sold in one year, and a royalty rate of 10% means an income of €10,000. You will need to decide whether that is worth the effort and any restrictions a licence agreement may impose.

Another way to work with a commercial software vendor could be a Knowledge Transfer Fellowship, often provided by a governmental funding body. This is ideal if your software will be a part of a commercial code or a new functionality in a software company's programme suite. A company always will prefer a commercial or permissive license, as they offer the best way to uptake external software. If your software is standalone it can be copyleft licenced, as the company can write an interface to their commercial code and they can charge a customer for that. When you approach a commercial partner with such a fellowship idea, discuss the problem they wish to tackle and then find the solution jointly. This will also help with the proposal that has to be written to get the grant in the first place. You are well advised to clarify expectations in the beginning so you can identify what you want to get out from this fellowship and what the commercial partner want to learn from you. You will certainly encounter how a full cycle of commercial software development is very different from academic software creation. This may help you, however with future software development.

8.1.2 Academic career and being a commercial software vendor

This combination allows you to wear both an academic and a commercial hat. You will need to tell your University that you have a second job outside academia and draw up the needed contracts. You may also want to partner with one or more likeminded persons to share the workload.

The idea of a company may have come about when customers prefer to deal with a software company rather than a private person or an academic institution. They pay for licenses, guarantee an income stream and expect you to provide support and new releases with new functionality. You will have to build up a support and maintenance infrastructure yourself and start at the beginning doing all by yourself. This will also decide how many customers you can accept. Once you can generate stable revenue streams with software sales, demo licenses and support/training you can look into more personnel.

To cope with both careers will be a bit of a balancing act. You could split your venture into an academic open source project, which allows to stay on top of your scientific domain and have students/post docs involved in the development of functionality. For your company you can use a more specialized commercial package of your software directly geared to solve your customer's problems. For example, this is done by SuessCo Simulations who have commercial products[15] and their corresponding academic open source software[16].

8.2 Business models for a commercial SWO

8.2.1 A non-profit organisation

A non-profit organisation (NPO) related to materials modelling software would sell its software (and services) and use any surplus generated (after running costs such as overheads and staff are deducted) to further the development and maintain the software.

You may want to check carefully the ownership of your software and if you owe royalties to your University or co-authors (in case they do not want to join your venture).

It is also wise, to build a network of people who actually would want to use your software before thinking about forming an NPO. Try to distribute your results and demonstrate features of your software to your potential market using international events.

You should decide on a name for your NPO and check if the domain is unique and people will find you easily and do not get misleading suggestions by search engines when they look you up on the internet. Spend some time to formulate your NPO's story well and provide your customer with a vision they can relate to. In a previous study (Goldbeck & Simperler, 2019) we asked software users what they value most. Insight into a particular problem and research had the highest value to them. They want their software to be robust, reliable and precise and they wish to trust their results. This should feature in your vision, too. Invest some funds into professional branding as your logos should look professional. You also should maintain a good marketing/sales webpage. Your customer should find out very easily what is new and where can they buy it. As an additional approach to increase your name visibility, you can offer sponsorship for focused conferences/industrial events, which adds your logo on their respective websites.

You will find more advice in the next chapter.

[15] http://www.suessco.com/simulations/

[16] http://www.magpar.net

8.2.2 A company

The first step of founding a company are similar to those of an NPO. Research in your respective countries, what formalities are needed to register a company, and check carefully the ownership of your software and if you owe royalties to your university or co-authors (in case they do not want to join your venture).

A company however is supposed to make a profit; i.e. there should be some money left after you have invested into your software. Similar to an NPO, you have to have a good branding, look after your customers, and generate income. Another import aspect is following:

You need to be professional and meet user expectations:

- Keep your software well maintained and follow Best Practises (Wimmer, Eyert, & Stokbro, 2018) when it comes to software development.
- Use version control as some customers will need to refer to particular software they have used. You may also have to keep earlier versions of your software, in case the industry you serve require that.
- Have a professional customer-relation management system in place.
- Have a professional licencing system in place. You need to be able to control who is using your software rightfully and who is using it illegally.
- Provide a good end-user experience with manuals and tutorials and offer support and training.
- Provide and update regularly your case studies that highlight the value that your software brings to the user.
- Provide regular releases with new functionality.
- Build your software modular as some users prefer to purchase only what they need rather than a full suite of tools.

You will have to charge for your software and think about pricing. You can orientate yourself on your competition to get an idea about orders of magnitude you can charge for. Your income should cover royalties (if you have to pay them), salaries, overheads, running the software development, preparing the training materials and documentation and there should be some money left to invest into your software.

Things you should charge for are

- The actual software via license fees
- Maintenance (often annual, gives access to new version and support)
- Services (Consulting, Training, Contract research)
- Configuration (if your software requires this)

Be aware that licenses can be sold regularly while services are often a one-off sale. Often services are bundled into a license sale, to make sure the user has sufficient knowledge to work with your software correctly. Ideally, if you offer test licenses bundle them with training to enhance the chance of a software sale. The user friendliness of your software will determine the best ratio here. In average, SWOs base their revenue on 20% services and 80% software sales.

Things you could charge for are:

- Visualisers, GUIs

- Pre- and Post-Processor
- In-house Databases

If there is a lot of free software on the market that covers these you may want to offer them for free, too. You need to research the market before you develop and invest in a tool that is not (easily) sellable.

Once you are up and running in your country and close geographical regions, you may wish to expand and build a global agent network. Agents are professionals that would sell your software on your behalf in their geographical location. They will usually take a cut for each sale they make and you have to negotiate with them a suitable arrangement. Agents are very beneficial as they speak local languages and are very familiar with local trade customs. Feel free, visit and tag along with them when they visit customers and organise training events with them. The latter are very important to nurture your remote customer base and will reflect in more sales.

You can use tools like SCAMPER and push towards new markets and diversification.

You have to look into the micro- and macroenvironment of the Materials Modelling Market (Kotler & Armstrong, 2017).

The microenvironment is closest to you and comprises your employees, customers, suppliers, distribution channels, competitors and the general public (Figure 3).

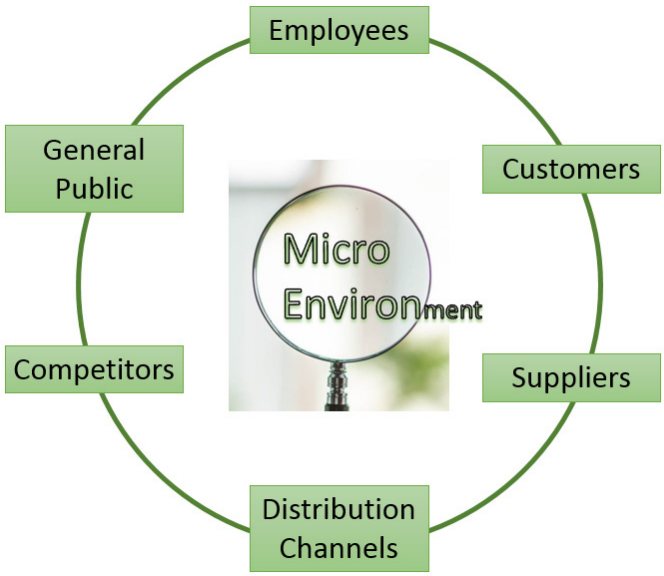

Figure 3. Six aspects of the microenvironment according to **(Kotler & Armstrong, 2017)**

- Employees: these are vital for your market as they develop, support and sell your software. Besides a passion for materials modelling and their technical skills, they need communication and high interpersonal skills.
- Customers: Find out why they buy your software; this will aid you how to market your software and services more targeted. You do not only sell software; you sell solutions to your customers that should seamlessly fit within their overall needs. Thus, your software becomes a product. You will

listen to your customers carefully and understand what it is that could bring value to what they want to achieve. If you have new functionality and software releases go proactively out to the customers and let them know. One should be prepared to spend a significant fraction of their revenues on marketing and sales. Finding good sales people is essential and difficult.

- Suppliers: You may or may not have suppliers. In the case of materials modelling software these could be academics who contribute to your software. You need to discuss your expectations early and involve them into a commercial way of developing software. If your contributor delivers a part of your code to late or not at all, this may change your release dates.
- Distribution Channels: These are agents or resellers, which act on behalf of you. Their marketing reflects on your business. You may want to work with reputable individuals so you can combine their reputation with your branding.
- Competitors: You need to watch their websites, social media, marketing campaign to keep an eye on what they are doing. Is there something you need to offer to keep up with them? Do you need to change your pricing? Do you get inspired, offer something better and get ahead of them?
- The general public: these are people you influence but who are not your direct customers. Can your software bring added value to society? For example, your software plays a role to aid with the construction of a greener motor, you could run campaigns via social media and point out the societal benefits. Go out and teach students and other communities to demonstrate your inclusivity.

Besides the microenvironment you also have to monitor the macroenvironment of your market. You cannot influence the macroenvironment but you have to operate within.

What can help here is the PESTLE[17] (often also referred to as PESTEL) analysis. The acronym is made up of Political, Economic, Social, Technological, Legal, and Environmental (Figure 4).

[17] Francis Aguilar is thought to be the creator of PEST Analysis. He included a scanning tool called ETPS in his 1967 book, "Scanning the Business Environment." The name was later tweaked to create the current acronym.

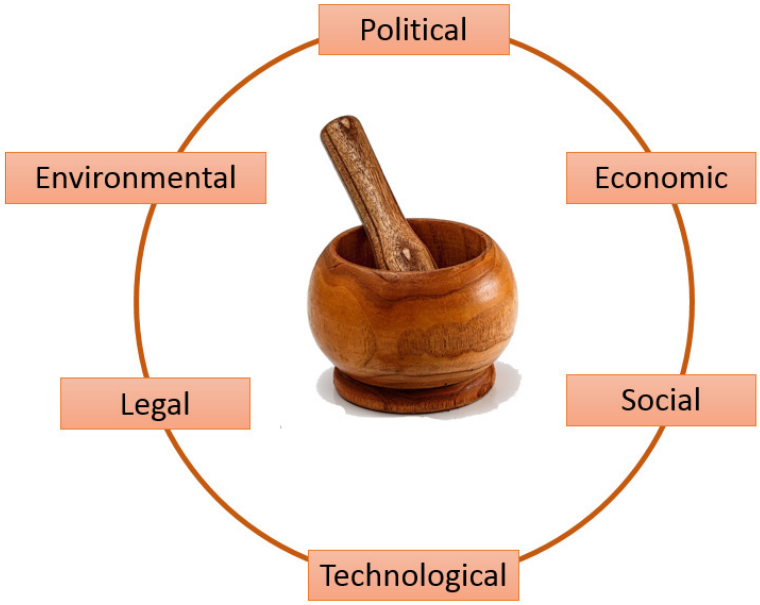

Figure 4. The six aspects of a PESTLE Analysis[16]

- Political: You have to follow your governments employment laws and laws in general. You have to pay tax. Governments can impose trade restrictions, which affect your software sales to certain countries. The education system of your government affects you as well as you draw your employees from there. Foreign employees may require visas and work permits, which you are responsible for. Countries you trade with may have different regulations, so you contact the Trade Commission, i.e. trade representations of your country set up in another one, to aid you.
- Economic: You should know about the economy of countries you trade with. If the economy becomes unstable (inflation, unemployment, etc.) people have other worries than to buy materials modelling software. Could you offer perpetual licenses to keep them going until the situation changes back?
- Social: Here you look at the workforce of another country and their general attitude towards learning new things. If a country puts a lot of emphasis on Materials Modelling in the University curriculum your software will be well understood. If not, you may think about how to become an enabler via training and offering “beginner packages” of your software and build on them.
- Technological: Your software requires hardware, databases, skilled IT personnel to install it, etc. Is this a given at your customer’s site? This will limit which of your products you can deploy.
- Technologies can change; thus, you always need to check if your software is fit for new technologies. Past examples were the advent of parallel computing and the usage of GPUs. You need to make sure your development follows the right trends, as you do not want your software to become obsolete or to invest too much money into the wrong trend.
- Legal: These factors overlap certainly with the political ones. It is very important that not knowing about laws (in your country or countries you trade with) does not exempt you from obeying them. Also, the wording of license agreements/contracts can vary from country to country. You are advised to have a legal advisor to help you with such things.

- Environmental: You as a company may have to follow environmental trends such as keeping a low carbon footprint by restricting air travel, printing on paper, watching your electricity use etc. However, Materials Modelling Software can aid to produce more environmentally friendly materials. If you sell software that describes a process, you can aid the circular industry, i.e. the cradle to cradle approach. Our community can bring value to this!

8.2.3 Funding for Business

Similar to open source SWOs you also may want to collaborate with industry, join EU projects or find grants that enable you to develop new functionality. There are funding schemas available that encourage companies to join international projects.

You can form consortia with manufacturing companies; for example, Accelrys Inc.[18], was running a Nanotechnology Consortium from 2004 to 2010:

The Consortium had 29 members and developed a substantial amount of new software solutions to meet customers' needs. However, consortia take a lot of time as you need to find partners and draw up contracts. You also have to carefully look into IP issues; if the collaboration leads to new software or new materials, who will own them? The consortium idea helped also in the foundation of Biosym Technologies[19] in the 1980's. To get started, the company launched a "Potential Energy Function Consortium" around 1985. This was followed by a polymer consortium around 1990 (over 50 members), a catalysis consortium, and an electronic, optical, and magnetic materials consortium (EOM). Cray Research launched the UniChem consortium (Zeyher, 1991)including a molecular DFT code and in 2012 Scienomics started an "Advanced Monte Carlo Simulations for Materials Properties" consortium[20].

You should consider that your customers may require funding to acquire your software. Your customers may encounter critique for buying commercial software when there is so much freeware out there. You should be prepared to aid with an argument why your software was the better solution.

8.3 Your Staff

You and your co-founders and colleagues will merge to a close-knit team of all-rounders to start with. It is very important you analyse what you and your staff are doing and when it is time to bring admin staff or specialists in. You need to find application-focused scientists, engineers and good managers. One sign is, that you get more work than you have personnel to do it. First of all, you need to check (as shown in the list) if you can afford a new head count and then who it is that you need:

- Your staff spends too many hours on expense sheets and their business travel booking ➔ hire an admin

[18] https://www.businesswire.com/news/home/20070709005195/en/Accelrys-Nanotechnology-Consortium-Delivers-New-Multi-scale-Computer

[19] Corning Inc. acquired Biosym Technologies in 1992.

[20] https://www.scienomics.com/sites/default/files/assets/resources/scienomics_mcc_announcement.pdf

- Your staff regularly overruns projects ➔ hire a project manager.
- Your staff spend time after regular work hours to support customers ➔ hire a technical support person.
- Your staff spend time for advising customers on pricing of the software ➔hire a sales manager.
- Your staff spends time after regular work hours to learn an exotic scripting language ➔ hire a short-term expert/consultant.
- Your staff spend much time to explain the software features to potential customers ➔ hire training and education specialists.
- Your workload and forecast thereof is too much for your existing headcount ➔ hire more staff.

Additionally, to have a specialist in experimental techniques and experimental data interpretation could bring an appreciable benefit.

Once your staff diversifies, you have to specify roles and have a curriculum for each of them. Your staff should have career progression with you and also develop their CV. If you get less sales or you change market you may have to reverse the hiring process and may have to make people redundant. It would help them immensely if they did acquire transferrable skills with you.

Besides staff related directly to your software, you may need admin staff that do paper work and travel arrangements to free your staff from non-billable work. If your finances permit, you may hire a person who is well versed with legal knowledge and can aid with contracts. Other relevant staff could be:

- Project Managers
- Distributor/ sales manager
- Software Tester
- Scrum Masters
- Solution Architects
- Translators
- Funding Specialists

8.4 Your Software

Your academic code should make the transition to software, a product you can sell. It is worthwhile, to bring rigorous software development in as early as possible. (Wimmer, Eyert, & Stokbro, 2018)
Some guidelines to get started can be found here:

- Software Sustainability Institute (SSI): https://www.software.ac.uk/resources/top-tips
- Guide to software development and projects at the Netherlands eScience Center: https://legacy.gitbook.com/book/nlesc/guide/details
- The Molecular Science Software Institute (MolSSI): https://molssi.org/education/best-practices/
- a good book on scientific software development in general:

https://www.amazon.com/Introduction-Performance-Computing-Scientists-Computational/dp/143981192X

- Some Blog Sites and Forums:
  http://best-practice-software-engineering.blogspot.com/
  https://softwareengineering.stackexchange.com/

The most common style for managing software development is Agile (http://agilemanifesto.org/) and more information can be found here https://www.atlassian.com/agile. This webpage offers some insight into the Scrum and Kanban methods (https://www.atlassian.com/agile/kanban/kanban-vs-scrum) with tutorials and background information. However, as one can see after further reading all these methods require a skilled person to manage a project successfully; thus, project manager and software developer for large projects should be ideally two different roles.

An interesting concept in Agile software development are Acceptance Criteria (AC). Microsoft Press defines Acceptance Criteria as "*Conditions that a software product must satisfy to be accepted by a user, customer or other stakeholder*." Google defines them as "*Pre-established standards or requirements a product or project must meet.*" AC are very useful in aiding with what should be developed first. It starts with the "user story" where a user, the community, or the RSEs have an innovative idea about software and what it should be able to do. Then the MoSCoW prioritisation method [21]can be used. The term MoSCoW itself is an acronym derived from the first letter of each of the four prioritisation categories (Must have, Should have, Could have, and Won't have (this time)). The method is a prioritization technique used in management, business analysis, project management, and software development to reach a common understanding with stakeholders on the importance they place on the delivery of each requirement; it is also known as MoSCoW prioritization or MoSCoW analysis.

Some further reading can be found here:

https://www.softwaretestinghelp.com/what-are-the-quality-attributes/

https://hackernoon.com/quality-attributes-in-software-architecture-3844ea482732

http://prince2.wiki/Quality

Your software tends to become bigger and more complex, so rewriting the code is not an option anymore as it is practically impossible. Communicate with your codevelopers often and plan wisely how to progress. Make your software easy to use. Explain the software potential based on industry relevant examples, offer training courses and provide support for pre- and post-processing.

Take into account that industry may have priority for confidentiality on some business sensible data and would like to keep key design solutions, new materials properties and important (realistic) results in house.

If a functionality is missing, think carefully if it is worthwhile to develop from scratch or if you rather should buy on in-license a 3rd party product.

[21] MoSCoW was developed by Dai Clegg of Oracle UK in 1994.

9. Outlook

The Materials Modelling Market is both exciting and challenging. Established SWOs have been serving this market for more than three decades now, which may be a challenging environment for newcomers. On the other hand, the materials modelling market is by no means mature. It is still dynamic, characterised by a large number of SMEs and significant mergers and acquisitions activity. There are a number of reasons for this market dynamics, including the still significant advances in key aspects of materials modelling, including the applied science of materials models, algorithm development which has been contributing strongly to increased speed and of course the increase in compute power and memory. In combination, these advances continue to push the boundaries of systems and challenges that can be addressed by materials modelling. At the same time, the need by industry to gain deeper understanding and use predictive tools is increasing as the low hanging fruits of materials and manufacturing development have been harvested already. Finally, the advance of machine learning and artificial intelligence is leading to a strongly increased 'hunger' for data, in particular consistent data that can serve as a basis for ML/AI. Materials modelling is unique in its ability to furnish those data and, in clever combination with experiments enable a much more rapid advance of ML/AI. There is already considerable activity in academia, for example in combining of physics-based modelling with data-driven and ML/AI approaches. Materials Modelling SWOs are needed to provide robust algorithms that for industry to benefit from these advances.

Hence imaginative approaches by newcomers are likely to be rewarded. They should learn where they can make a difference with their software and use a "Blue Ocean Strategy"[22] (Kim & Mauborgne, 2015) to create an uncontested market space and new demand. This makes the market very exciting as in turn the longer established SWOs have to be inventive, too, to not become a commodity.

The present and future SWOs have to monitor the market carefully, as it is very complex. In future, it should not be seen as black and white, i.e. a simple division into continuum and discrete modelling market. There will be grey shades, as there is a growing demand for software that can handle or contribute to solving multi-scale and multi-physics problems. Industry is going circular, which means instead of looking into business processes leading from cradle to grave, they are looking for cradle to cradle. A product is not designed anymore for going to waste, it shall serve as resources for a new one. To model such processes the knowledge of the chemical composition and behaviour thereof becomes pertinent and will require workflows that cover discrete and continuum models. Also, from a modelling perspective, continuum experts see that their models improve if they use more information from discrete modelling. Thus, present and future Software Owners must look into ontologies and metadata schema to enable interoperability and achieve a 'digital continuum'. The EMMC has been active in this field, leading to the recent release of the European Materials & Modelling Ontology (EMMO) (Friis, et al., 2019).

The SWOs must also keep an eye on the development of hardware. How will the software landscape change with the advent of novel HPC architectures and Quantum Computing, the latter having early applications in quantum chemistry, for example as reported by (Budde & Volz, 2019). Also, user interfaces keep on evolving,

[22] Red and blue oceans are used to denote the market universe. Red oceans are all the industries in existence today – blue oceans denote all the industries not in existence today.

with the ever-growing use of handheld devices and the advent of Virtual Reality, further changes in usage patterns of materials modelling are to be expected.

Other important trends from the consumer domain that will most likely change the landscape of materials modelling software offerings are the increasing trend towards on-demand services (e.g. music streaming) and the change in product and service distribution models towards large scale marketplaces (e.g. Amazon).

These changes could in particular benefit SMEs that often do not have the infra structure to permit them to do materials modelling, i.e. they lack the resources for hardware and software acquisitions (or long-term licensing arrangements). Cloud-based and SaaS offering could provide a means to close that gap, in particular as security concerns regarding cloud are receding. Digital marketplaces could in the future offer such integrated services and be a means for the still quite scattered materials modelling ecosystem to become more homogeneous. In any case, there will be plenty of opportunities for SWOs in this field.

10. Acknowledgements

The authors and the EMMC are grateful to all members of the EMMC-CSA project and Software Owners and Software Users who took part in the meetings, the interviews and the online surveys and their corrections and suggestions of the White Paper. Their effort made this work possible.

The work has received financial support from the EU H2020 project EMMC-CSA GA n. 723867.

11. Glossary of terms and abbreviations

BSD – Berkley Software Distribution

CAD – Computer Aided Design

CAE - Computer Aided Engineering

CATWOE - Customers, Actors, Transformation, Worldview (“Weltanschauung”), Owner, and Environmental Constraints.

EMMC – European Materials Modelling Council

GPL - GNU General Public License

GUI – Graphical User Interface

HPC – High Performance Computing

IP – Intellectual Property

LGPL - GNU Lesser General Public License

MIT – Massachusetts Institute of Technology

MoSCoW - Must have, Should have, Could have, and Won't have (this time)

NPO – Non-Profit Organisation

PESTEL - Political, Economic, Social, Technological, Environmental and Legal.

RoI – Return on Investment

SaaS – Software as a Service

SCAMPER - Substitute, Combine, Adapt/Adjust, Modify, Put to alternative use, Eliminate and Reverse/Rearrange

SME – Small and Medium Enterprises

SWO - Software Owner

SWU – Software User

12. References

Ansoff, H. I. (1957). Strategies for Diversification. *Harvard Business Review 35*, 113-124.

Bergvall-Kareborn, B., Mirijamdotter, A., & Basden, A. (2004). Basic principles of SSM modeling: An examination of CATWOE from a soft perspective. *Systemic Practice and Action Research 17*, 55-73.

Budde, F., & Volz, D. (2019, July https://www.mckinsey.com/industries/chemicals/our-insights/the-next-big-thing-quantum-computings-potential-impact-on-chemicals). *The next big thing? Quantum computing's potential impact on chemicals.* Retrieved from McKinsey & Company.

Eberle, R. F. (1972). Developing Imagination Through Scamper. *Journal of Creative Behavior 6*, 199-203.

Friis, J., Løvfall, B., Ghedini, E., Goldbeck, G., Schmitz, G., & Hashibon, A. (2019, August 13). *EMMC.* Retrieved from Documentation of EMMO: https://emmc.info/wp-content/uploads/2019/08/EMMC-CSA_-D2.7_M232_vfinal_PU-WEB.pdf

Goldbeck, G. (2012, May 1). *The economic impact of molecular modelling of Chemicals and Materials.* Retrieved from Zenodo: http://doi.org/10.5281/zenodo.44359

Goldbeck, G. (2017, January 27). *The scientific software industry: a general overview.* Retrieved from Zenodo: http://doi.org/10.5281/zenodo.260408

Goldbeck, G., & Simperler, A. (2019, January 16). *Business Models and Sustainability for Materials Modelling Software.* Retrieved from Zenodo: doi:10.5281/zenodo.2541723

Hanna, R., & Weinhold, I. (2017). *The Demokratization of CFD.* Retrieved from MENTOR GRAPHICS CORP.: https://go.mentor.com/4RREr

Kim, W. C., & Mauborgne, R. (2015). *Blue Ocean Strategy, Expanded Edition: How to Create Uncontested Market Space and Make the Competition Irrelevant.* Boston: Harvard Business School Publishing Corporation.

Kotler, P. T., & Armstrong, G. (2017). *Principles of Marketing, Global Edition.* Harlow, United Kingdom: Pearson Education Limited.

Smyth, D., & Checkland, P. (1976). Using a Systems Approach: The Structure of Root Definitions. *Journal of Applied Systems Analysis 5*, 75-83.

Wimmer, E., Eyert, V., & Stokbro, K. (2018, May 4). *White paper for standards of modelling software development.* Retrieved from Zenodo: https://doi.org/10.5281/zenodo.1240212

Zeyher, A. (1991). Cray Constructs Powerful Chemistry. *Computers in Physics 5*, 369-371.