The scientific software industry: a general overview

Author: Dr Gerhard Goldbeck

Goldbeck Consulting Ltd, St John's Innovation Centre, Cowley Road, Cambridge CB4 0WS, United Kingdom
http://www.goldbeck-consulting.com

The financial support from the Durham University in the production of this report is gratefully acknowledged.

Publication Date: January 2017

For further information, contact: info@goldbeck-consulting.com

Contents

Introduction

The remit of the current report is to support the exploitation of scientific software developed in academia by providing an overview of the relevant software industry as well as considerations concerning licensing and commercialisation of scientific software.

Particular focus and examples are given for modelling and simulation software that is based on a fundamental chemical and physical description of matter, i.e. materials modelling [1]. The use of such modelling technology and its industrial impact has been elaborated in separate reports [2,3].

The aim is to provide a high level overview of the industry, its typical requirements and business models, and in particular to address the following topics:

1. The structure of the software industry.
2. Requirements for software development: both in-house and through collaboration.
3. Routes to market for scientific software, e.g. via software houses or direct licensing into specific industries.
4. Commercialisation requirements for software: standards, IP ownership, licensing schemes.
5. Warranty and liability issues.

Industries and markets served

The relevant industry sectors are those that depend strongly on innovations involving scientific know-how, including:

- Pharmaceutical and biotechnology
  - Discovery (“Life Science”)
  - Development (“Materials Science”, Analytical Chemistry, Process Chemistry)
- Chemicals industry
- Materials development for a wide range of products, such as automotive, aerospace, and other transport applications, consumer packaged goods (home and personal care, foods), adhesives, packaging, plastics etc.
- Electronic devices

A closer look at the innovation process in general is helpful in order to understand where scientific and related engineering software can add value, and how large the different sectors are.

As illustrated in Figure 1, as the innovation process moves from fundamental exploration and discovery towards the product, the technical risk decreases, but in general the associated resource requirements, e.g. pilot plants, lead to an increase in cost. A typical figure underpinning this argument was reported in the 2003 OECD Science, Technology, and Industry Scoreboard. In developed countries with high R&D intensities, basic research is less than one fifth of total R&D spending [4].

Not surprisingly, the number of software users is also larger downstream towards engineering and manufacturing. Note though that the technical risk is higher at the start of the process, hence each user can potentially add significant value by reducing that risk, and the potential costs involved with failure downstream.

(a)

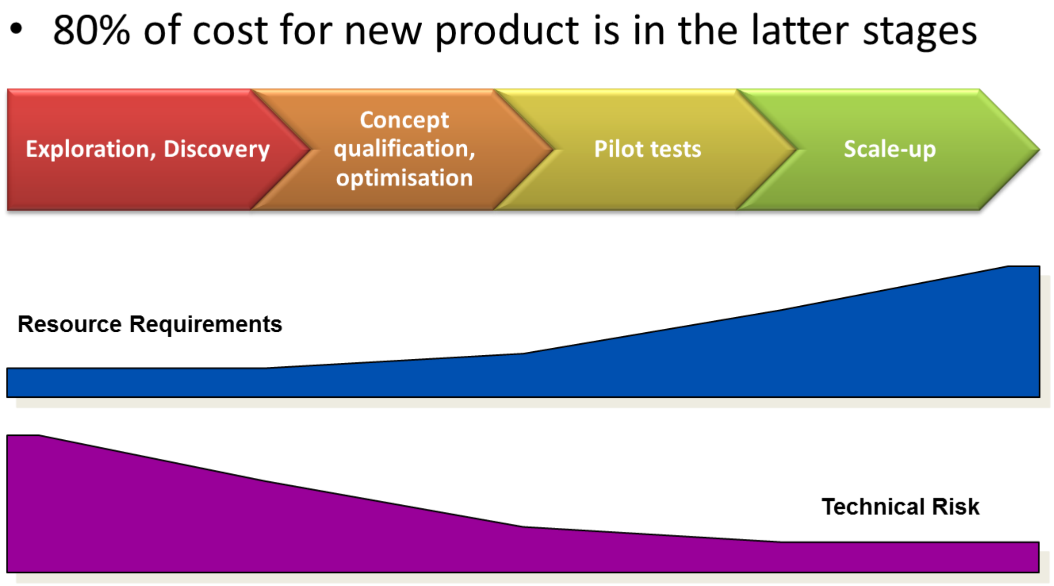
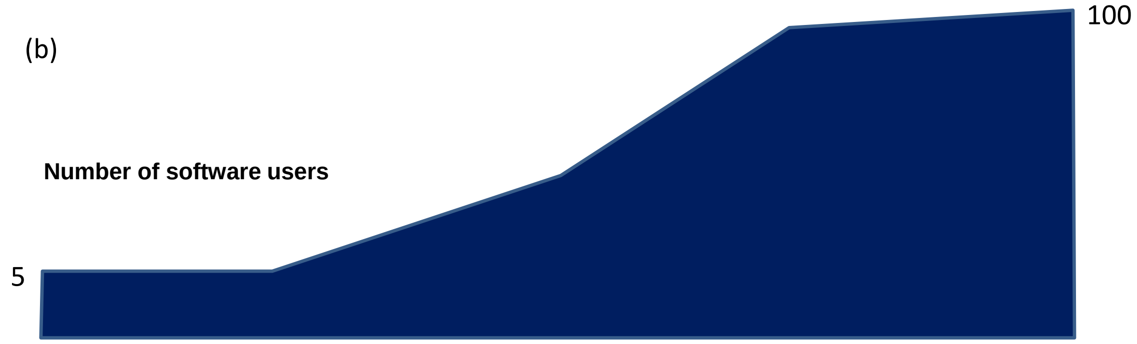


Figure 1: (a) Stages of the R&D and Innovation process and related resource requirements and technical risks; (b) The number of software users increases from the early R&D to the latter engineering stages.

(b)



Market size

The size of the software industries serving the different industry sectors and the various segments along the innovation chain can roughly be estimated in the following way:
As a starting point, one can take the R&D spending of an industry sector as a whole. A good source is the R&D Scoreboard published by JRC (European Commission)
http://iri.jrc.ec.europa.eu/scoreboard.html
For example:

- Pharma and biotech combined R&D spending is about $100bn.
- Chemicals, including oil and gas: about $50bn.
- Electronics: about $30bn.
- Aerospace and defence: about $20bn.

For software supporting the early parts of R&D, experience and industry analysis has shown that the spending on scientific software is in the range of 0.1% of total sector R&D spending. For example

- For pharma and biotech, the combined R&D spend is about $100bn, and the spending on scientific modelling and simulation software is about $100m.
- In the Chemicals/Materials industry the R&D spending is about $50bn, and the molecular modelling software market for chemicals and materials is about $50m.

The number of users of scientific software such as molecular modelling in chemicals/materials is in the range of a few thousand in industry and a few tens of thousands in total.

For software supporting the later stages of the innovation process there are typically 100 times as many users and at least a factor 10 in market value. It is acknowledged that these are rough figures, but they typically serve well for a first assessment of potential markets. High quality market data are rarely available due to the cost involved in such market research activities.

Structure of the science and engineering software industry

In terms of applications and markets, scientific software supports research and development in industry as well as academia. It provides value by enabling researchers to widen the scope of their research, get new insights that can lead to novel products, reduce efforts and costs associated with the synthesis of new chemicals and materials, make more efficient use of experimentation and improve the interpretation of results [2,3].

In order to categorize the structure of the software industry it is useful to consider different angles such as the markets served, the type of application and type of user:

- Markets served:
    - Materials science and life science markets
    - Scientists versus engineering users
- Models applied:
    - Discrete chemistry and physics based models: i.e. electronic, atomistic and mesoscopic
    - Continuum physics based models
    - Data-based models
- Informatics software, including ELNs and LIMS
- Goals and value of commercial software
- Historical development and maturity of industry

Software types/user segments

Regarding software types, one can distinguish between software supporting the different stages from basic research to manufacturing, but also between physics/chemistry based modelling and informatics (date-based) software.

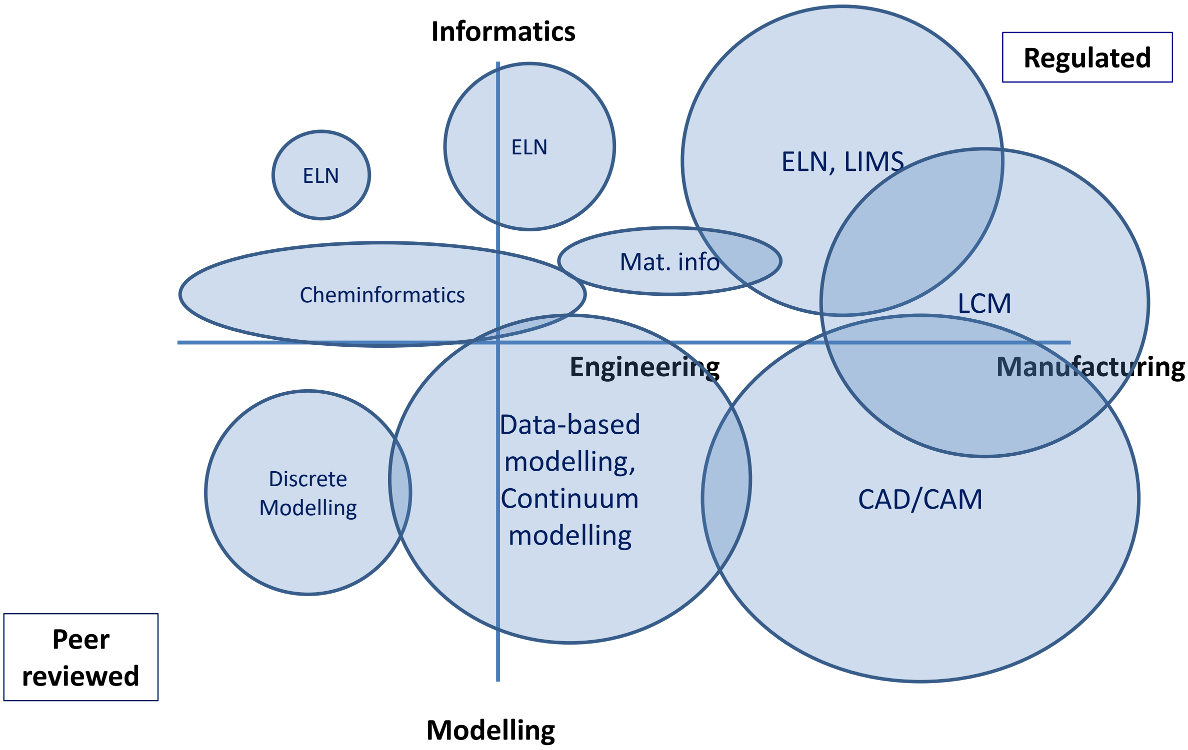


Figure 2: Different types of software from basic research to manufacturing, and modelling to informatics. The size of the ellipses is a rough indication of the market size, but not to scale. Abbreviations used: ELN: Electronic Lab Notebook, LIMS: Lab Information Management Systems, LCM: Life Cycle Management, Mat-info: Materials Data and Informatics software, CAD/CAM: Computer Aided Design/Manufacturing

The software industry is traditionally segmented into companies that specialise in some way, for example along industry sectors (Life Science, Materials Science), user types (modeller, bench chemist, engineer), or methods (chemistry based modelling, physics based modelling, engineering modelling, instrument data processing, informatics etc.). In recent years there has been some integration and consolidation along the innovation value chain, e.g. with the takeover of Accelrys by Dassault Systemes, of e-Xstream by MSC software and of CD-Adapco and the by Siemens PLM . The goal is more and more to capture the whole product life cycle and to accelerate the innovation process shown in Figure 3. Scientific software mostly applies to the initial stages but has also been instrumental in trouble shooting, in particular in cases where the depth of knowledge of the underlying chemicals, formulations and materials was found to be insufficient during upscaling.

Goals and value proposition

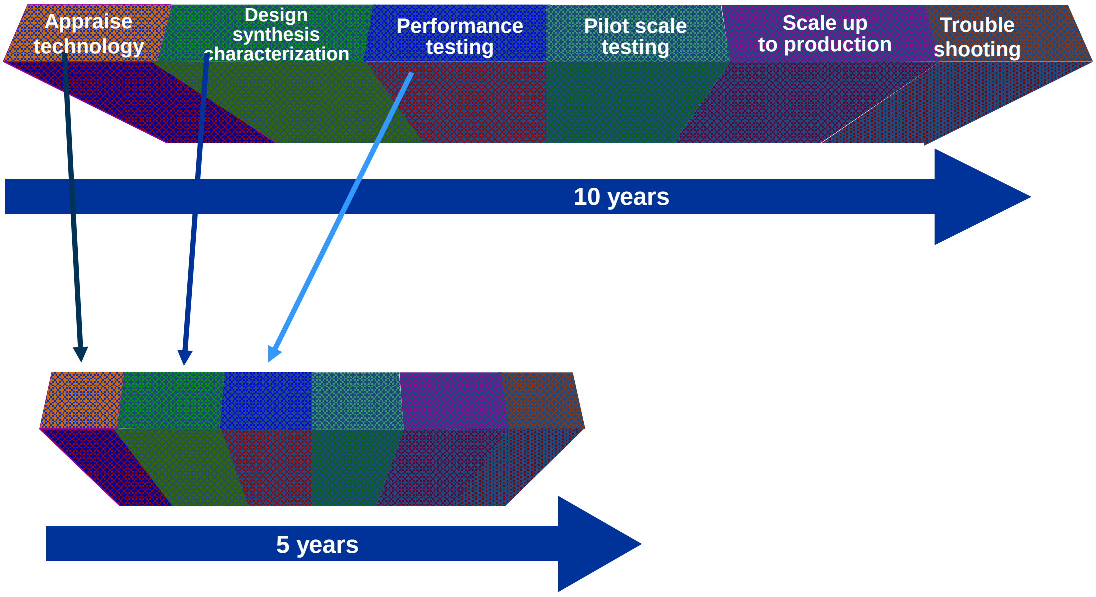


Figure 3: Innovation Process Acceleration: scientific software in particular impacts the early stages, but a thorough knowledge base also improves upscaling and reduces the trouble-shooting time.

The value proposition in general is to reduce overall development time. In the early stages, modelling improves the ability for creative exploration, and then focus the development on the most promising candidates as shown in Figure 4. In this way, the number of costly late stage development projects is reduced and their potential for failure is minimized.

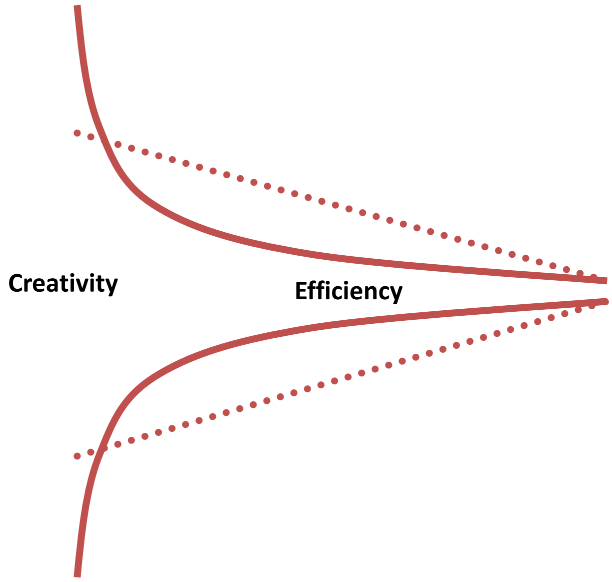


Figure 4: Change in the innovation funnel by applying modelling (from dotted to solid line). The early stage, creative exploration is enhanced, leading to increased efficiency and effectiveness.

Key impacts that have been quoted in previous reports [2,3] include:

- More efficient experimentation
- Broader Exploration and Deeper Understanding
- Saving a Product Development Project and/or Accelerated Product Development

More specifically, the software industry supports the workflow of the scientists/modeller:

- Task 1 supported by training, help systems, consulting.
- Tasks 2, 4, 5 are where commercial software adds value, typically not well covered by academic and free software.
- Task 3 is where powerful new codes are sought by industry.

Integration into software packages aims to minimize the "time to solution", including preparation time, effort to determine input parameters, CPU time, post-processing and interpretation of results. Anecdotally at least it does that very well, reducing the time spent by the user by a factor of 5-10.

Chemistry based modelling and simulation software industry

A key case in the scientific software industry is that based on discrete modelling technology. It is characterized by the following:

- Market size: about $150m
  - Largest sector: life science/drug discovery (~ $100m)
  - Materials science, about $50m.
- Companies
  - Life Science: some companies (including Biovia, Schrodinger and CCG) of similar importance covering a range of applications each plus niche players that integrate with the large companies in some way (ISV or similar). Strong focus on integration.
  - Materials Science: one large company that includes a wide range of applications (Biovia), plus a large number of smaller companies (including Schrodinger, Materials Design, Scienomics, QuantumWise, CULGI). Less integration.
- Many academic codes and users that don't license commercial software.
  - Biovia revenue market share: 50%, but user share about 5%!

The level of maturity of the industry can be illustrated by the Gartner Hype Cycle (Figure 5).

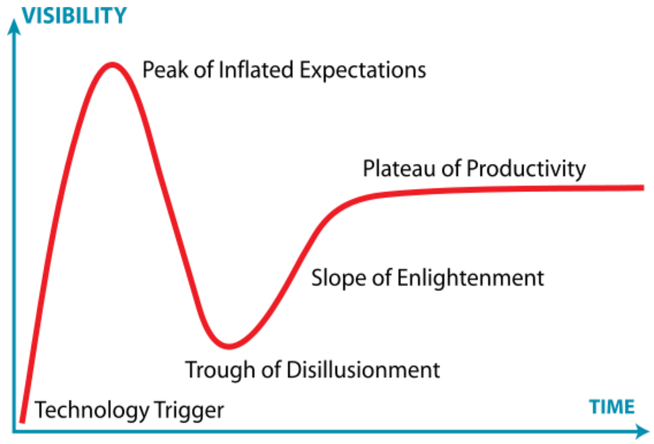


Figure 5: Gartner Hype Cycle. Ref: Jeremy Kemp at en.wikipedia, see http://en.wikipedia.org/wiki/Hype_cycle

The Technology trigger of the industry dates back to 1980s with the availability of suitable theories, codes and graphics workstations. Investors backed software companies leading to a peak of expectations and prices of software installations of about $100k per user. Following disillusionment with the results due to a mixture of inflated expectation, poor integration of simulation with experiment and lack of compute power we are now in the 'slope of enlightenment phase' where:

- More can be done at lower cost of computing.
- There is better integration with other activities.
- More users (though especially in academia still).
- Commercial price pressure: price per user $30-50k.
- Industry conservative re costly in-licensing agreements!

Software commercialisation, innovation and business

Considering the way in which software is offered to the user ("packaged") one can distinguish between the following three approaches to commercialisation:

1. Highly integrated, all proprietary packages.
    a. Integrate many different pieces of software to cover a wide range of applications and use cases.
    b. Integration is proprietary: can only be done by software house.
    c. Requires LDA (license-distribution agreements) with the original software owners.
    d. Example: Biovia Materials Studio
2. Integrated, partly open source packages.
    a. Set up for quick integration of external applications.
    b. Mixture of proprietary (User Interface, some apps), and open source.
    c. Example: Scienomics MAPS
3. Workflow and data pipelining frameworks.
    a. More oriented to support workflow, than graphical UI
    b. More flexible for integration and customization.
    c. Supports ISV partnerships (see below).
    d. Examples: Biovia Pipeline Pilot, KNIME, Esteco modeFrontier

The development at a software provider is directed towards focusing on the core expertise and distinctive competence of the company, which may lie in the integration, user interface or IT framework infrastructure.

In-house development teams focus on software engineering:

- User interface functionality and performance.
- Integrating new developments from third parties, e.g. academic partners.
- Core software development. What is core varies by company and over time. There are only few examples where the scientific developments themselves are at the core of the company activity.
- Keeping software up to date with hardware and operating systems developments.
- Documentation.
- Quality control.
- Release engineering.

Product managers are responsible for identifying market problems, technology assessment including collaborations with academics that can address market needs, roadmaps etc., see: http://www.pragmaticmarketing.com/pragmatic-marketing-framework

Typical routes for developing new software products to meet certain market problems include

- In-house development
    - Focus on integration, user interface, pre-/post- processing, workflow, i.e. tasks 2, 4, 5 in above schematic
    - Not usually done for the actual models (Task 3): expensive, long term, deep science required
- In-licensing of academic codes
    - Based on business case (see below): need very good fit
    - Term >15 years
    - Sold as integrated package by software company
- Integration nodes to free/open/independent codes
    - Company just charges for User Interface
    - Get more features without taking full responsibility for product

If a new technology is identified for potential in-licensing, the product management function at a software company will have to demonstrate that there is a business case for licensing and integrating new technology. The case will cover issues such as

- Strategic fit with company objectives and distinctive competence
- Addresses significant market problems
- Proven technology? Risks understood?
- Competitive strategy in place
- Profit and loss analysis very positive over 5 years
- Adds significant growth to company revenues
- Fits with company sales channels and strategy
- Fits with support skills and framework.

Since the barrier to in-licensing of a new technology by a software company is quite high, and integration into a comprehensive package relatively inflexible, the model of independent software vendor (ISV) integrations into more open and flexible frameworks has been growing. Basically, the software company concentrates on building platforms that allow integration of many products. The aim is to make the framework a de facto IT standard in a sector by:

- Powerful platform technology
- Covering all features a user might need by integrating products from a range of sources: ISVs

The ISV is a separate company that is responsible for its own product and business regarding licensing, support, warranty, liability etc. The relationship with a platform provider enables the ISV to obtain access to new markets. Typically, the platform integrator charges the ISV for the integration.

Examples include:

- http://www.ansys.com/About-ANSYS/partner-network/software-partners
  http://accelrys.com/partners/our-partners.html
- http://www.esteco.com/home/mode_frontier/Integration.html

Routes to market for scientific software

The main avenues for scientific software and its results to get to the end user are via

- License and distribution agreement with a software company
- Direct distribution, either from the university or using a spin-off company as the vehicle
- Using consultancy and contract research to get the 'solution' to the end user rather than distributing the software

Regarding a license/distribution agreement with a software company there are different types of agreement regarding exclusivity and related royalties, including

- Exclusive for all sectors (industry, academic).
- Exclusive for some sectors, e.g. industry only.
- Non-exclusive (seeking similar agreements with multiple vendors).

Royalties typically range from 5-20% of revenue depending on

- Level of exclusivity: non-exclusive range: 5-10%, exclusive range 12-20%
- Higher end will only be paid if there is continued support of the code and on-going development which result in innovative updates.

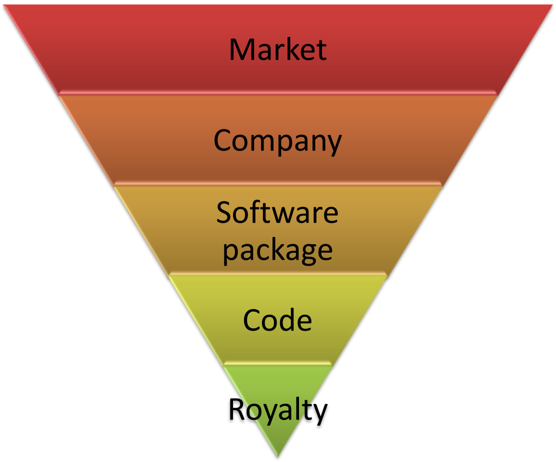
Advantages of this type of relationship are:

- Benefit from marketing efforts of software company.
- Avoid having to handle sales, licensing, and support.
- Benefit from additional testing and bug fixing done by the software company.

On the other hand, it means

- Being tied in to one partner for many years.
- Hard to get out if things don't work out…
- Additional demands on academics (bug fixes, speak at customer meetings, etc.)

- Only get a percentage, is it worth the effort?



- Market: $50m
- Company: $20m
- Package: $10m
- Code: $0.2 – 2m
- Royalty: $30 – 300k

- Note that at lower end it may be counterproductive to go for an exclusive in-licensing relationship!

Figure 6: Inverse Pyramid from Market Size to Royalty payments for a piece of software.

Another option is to go direct to the market by setting up a spin-off company. It provides a vehicle to license and distribute software both direct and via ISV partnerships with larger software companies. A direct end user license would typically use a simple pricing model, say free or a relatively small fee for each academic license, and a few thousand Euros/Dollars per industry license, providing a perpetual one-time license without any update or support. Updates therefore require a new license, and support is typically provided by the user community itself via a forum or wiki.

Further distribution can be achieved by seeking an ISV relationship with a software house.

- Pros:
  - Retain independence: ability to set price and retain full revenue
  - Benefit from potential tax advantages for SMEs
  - Ability to develop a support and contract research part of the business
  - Potential to sell company to a software house at a later stage
- Cons:
  - Financing requirements
  - Costs associated with running a business, including marketing, sales, legal

Finally, it is worth considering consultancy and contract research business models. There is significant demand due to increasing outsourcing even in R&D. Advantages are that it

- Avoids costly ownership of software
- Avoids costly commercial software development
- “Expert tools in the hands of expert users”

The key disadvantage is that the business does not scale as well as software, since each new customer also requires substantial investment of time to carry out the project.

Examples of (small) companies spun out of the academic sector are:

- http://www.ceposinsilico.de/
- http://www.simune.eu
- http://creative-quantum.eu/
- http://www.molecular-dynamics.de/

Commercialisation requirements

Software standards

The requirements on software code for commercialization reflect the innovation workflow and level of regulation of the industry sector and the section within the workflow. Typically, early stage innovation is more exploratory hence less regulated and likewise the tools and procedures used are also less regulated. On the other hand, in fields such as pharmaceutical research using software tools such as Electronic Lab Notebooks that are used as part of regulatory filing procedures, the software including all software tools (such as modelling) used as part of the system need to comply with stricter regulations.

These regulations affect the procedure used in the code development, review, and quality assessment procedures of the software company, more so than those of the academic groups that may have provided some of the underlying software. Nevertheless it is important for academic groups interested in commercialising their software and developing a longer term, successful relationship with a software company to be aware of these requirements, and to support them as much as possible. The basic requirements of a software company regarding standards are

- Scientific quality (as demonstrated in peer review publications).
- Generality (not just applicable to a small range of systems of academic interest).
- Technical reproducibility and robustness.
- Set of QA tests that must be met when changes are made.
- Readable and commented code, which is easy to maintain.
- Using recent coding standards (but a wide variety of languages are typically acceptable).
- If possible HPC ready.
- Maintain and use a common code repository.
- Distinguish between release and development versions.
- Procedures for making changes in development version.

For further, detailed discussion, see the EMMC guidance on quality assurance for academic materials modelling software engineering [5] and the online "Primer on software development best practices for computational chemistry" [6].

Intellectual property issues

Since a piece of software is not a static product, issues relating to the ownership and distribution rights to any developments following the date of the agreement, and during the term of the agreement need to be clarified. In some cases any development is excluded from the license, but there may be an understanding that the license agreement is updated on a regular (e.g. annual) basis. In most cases

- The licensee has the right not only to use but also to modify the software. The IP in these modifications may be kept by the software house, but typically a perpetual use/modify license is given back to the licensor to enable further joint development.
- The distribution rights to any further developments, or "improvements" made by the licensor are granted to the software house as part of the agreement.

Regarding such further developments ("improvements") the licensor may in some cases retain the right to determine the terms on which any such improvement is provided. For example, a license may have been granted on an exclusive basis for a piece of software, but further developments could include code that is distributed on the basis of public licenses. This makes sense only if such code is in some way standalone. An example would be a general mathematical or simulation procedure, such as a geometry optimizer. Also, in some cases an attempt is made to differentiate updates and improvements to a code and substantially new software development.

Wikipedia provides a very good overview of software licences and licensing models [7].

Warranty and liability issues

If code is licensed to a software house, the licensor (e.g. the University) is typically asked to warrant at least the following:

- The software substantially complies with a register of specifications and requirements typically provided as an Appendix.
- Licensor either owns copyright in or has the right to license the software.
- In case that the software contains parts to which the Licensor has no distribution rights, including public domain components, these must be clearly identified in a register, and the software company will need to be satisfied that either the licence is compatible with commercial distribution [5,7] or that these can either be removed, replaced without affecting the commercial viability of any intended software product.
- To the best of licensor's knowledge the software does not infringe copyright nor infringe any patents.
- Licensor will retain and have retained all patents, copyrights, trade secrets and other proprietary rights.

In addition, there are typically general legal clauses that warrant that there is no conflict with other agreements, or misuse of confidential information etc.

There may also be explicit disclaimers regarding

- Patents (e.g. expressly stating that there was no requirement of a patent search)
- Implied merchantability or fitness for a particular purpose.

In case of some level of exclusivity of the licensing and distribution agreement, there will be clauses warranting that the licensor has not granted such rights to others.

Indemnity

Licensor will be required to indemnify licensee against losses, damages, expenses and claims that arise as a result of a breach of the warranties. There is typically no limitation of liability regarding warranties relating to copyright, proprietary and licensing rights. However, there is typically a limitation of liability relating to warranties on specifications, and any claims arising in relation to that. These are typically capped at the actual level of fees obtained. No liability is assumed for damages suffered as a result of using the software including lost profits, data, computer failure etc.

The above warranty clauses reflect the licensing and distribution terms and software license agreements that are entered into by the software companies with their customers. These will include clauses that maintain the rights of the company to license the software, a warranty that the software performs to specifications (typically provided in a User Manual), and that in case of reported defects regarding these specifications, reasonable efforts will be used to fix the reported defects within a certain timeframe (60 – 90 days, next release etc.). Regarding these defects, the liability is typically limited at the amount paid by customer for the license.

The software company, whether in science or engineering, does not assume any liability for any damages suffered as a result of using the products in any way.

ISV relationships

In case of an independent software vendor relationship, the distribution remains with the ISV, and hence no rights nor warranties and liabilities are assumed and need to be determined between the ISV and the software company. However, as an ISV may want to distribute its software together with that of the software companies to prospective users for evaluation purposes, the ISV will be responsible and will be asked to indemnify the software company for any claims resulting from such distribution.

Industry Trends

Integration has been the key trend in the industry for many years. However, the meaning of the term and related developments has shifted from integration within sub disciplines (such as lab integration, multiscale modelling integration etc) to a much broader and wider integration to support future digitalisation and 'Industry 4.0' development. Some of the vision was already expressed in the 2008 "Integrated Computational Materials Engineering" (ICME) report [] which claimed that a faster insertion of new material into actual product can be achieved by "integrating computational materials science tools into a holistic system that can transform the engineering design optimization process." It stated that "Integration is viewed differently in each of the communities expected to contribute to the growth of ICME" and that "while this [multiscale integration] concept is often useful for defining a modeling strategy, its importance is sometimes overemphasized".

As already mentioned in the section on Software types/user segments, there is more and more integration along the value chain, building Product Life Cycle solutions that reach all the way back to the materials/formulations/chemicals. The value placed on such solutions by large manufacturing companies is exemplified by a Boeing Corporation patent for a "Product Chemical Profile System" which is able to pull together and query all levels of information about a product down to the chemistry level, integrating models, simulation and analysis [8].

In order to deliver on the wide range of requirements, there is a trend towards a more flexible type of integration [9] enabled by frameworks with open Application Interface standards, which can support cloud based software infrastructures and open up the possibility for digital market places in scientific software. In general this development should make it easier for start-ups and ISVs to bring their software to the end user, and vice versa for the end user to access academic developments more quickly and easily. Successful early examples are the Nanohub: http://nanohub.org/; which has seen a rapid growth in software applications and software users. The manufacturing-hub: http://manufacturinghub.org/ is geared to bringing such applications to manufacturing sector companies that otherwise don't use modelling software. In Europe, projects that will extend this 'hub' concept to online marketplaces will soon be under way [10].

References

[1] What makes a material function? Let me compute the ways. European Commission, Directorate-General for Research and Innovation (2015), DOI: 10.2777/23288

[2] G. Goldbeck, The economic impact of molecular modelling, (2012), https://doi.org/10.5281/zenodo.44359

[3] G. Goldbeck and C. Court, The economic impact of materials modelling (2016), https://doi.org/10.5281/zenodo.44780

[4] OECD, Science, Technology and Industry Scoreboard. Paris, France (2003).

[5] European Materials Modelling Council; EMMC guidance on quality assurance for academic materials modelling software engineering (2015), https://doi.org/10.5281/zenodo.192025

[6]J. D. Chodera, P. B. Grinaway, A. Rizzi, G. Ross, L. N. Naden, A. S. Rustenburg, Software Development Best Practices for Computational Chemistry, Choderalab, accessed Dec 2016 via https://github.com/choderalab/software-development

[7] Software license, https://en.wikipedia.org/wiki/Software_license

[8] The Boeing Company, Product Chemical Profile System, Patent WO2015060960 (2014), https://materialsmodelling.com/2016/02/08/boeing-is-moving-ahead-with-integrating-chemistry-and-materials-modelling-into-the-product-life-cycle/

[9] European Materials Modelling Council, EMMC Roadmap for materials modelling (2015, 2016), https://emmc.info/roadmap/

[10] "Next generation system integrating tangible and intangible materials model components to support innovation in industry", Call for proposals in the EU H2020 2016-2017 Work Programme Part: Nanotechnologies, Advanced Materials, Biotechnology and Advanced Manufacturing and Processing; http://ec.europa.eu/research/participants/portal/desktop/en/opportunities/h2020/topics/nmbp-25-2017.html